PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-311004
(43)Date of publication of application : 07.11.2000

(51)Int.Cl. G05B 19/02
H04L 12/40

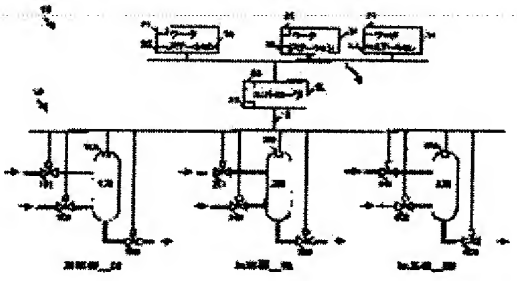
(21)Application number : 2000-069117
(22)Date of filing : 13.03.2000

(71)Applicant : FISHER ROSEMOUNT SYST INC
(72)Inventor : IRWIN WILLIAM G
HAVEKOST ROBERT B
STEVENSON DENNIS L
DEITZ DAVID L

(30)Priority
Priority number : 99 267431 Priority date : 12.03.1999 Priority country : US

(54) INDIRECT REFERENCE IN PROCESS CONTROL ROUTINE

(57)Abstract:
PROBLEM TO BE SOLVED: To attain reduction of necessity of a controller memory by generating an instantiated version of a control routine and controlling a unit module by performance of instantiated version of a general control routine.
SOLUTION: In a controller 12, a unit object indicates a phase theory which is independently operated by the controller 12 on a unit module and is potentially in parallel to other active unit phases simultaneously on a different unit module. A unit phase object is one instantiated version of a phase class which is decided by using an areas decision table for the unit module to which the unit phase object belongs. Then, the controller 12 actually performs the unit phase object (an instantiated version of the phase class during an execution time) and leaves a general version of the phase class in a memory 22.



LEGAL STATUS

[Date of request for examination] 26.02.2007

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2000－311004
（P2000－311004A）
(43)公開日　平成12年11月７日(2000.11.7)

(51)Int.Cl.[7]	識別記号	ＦＩ		テーマコード(参考)
Ｇ０５Ｂ　19/02		Ｇ０５Ｂ　19/02	Ｔ	
Ｈ０４Ｌ　12/40		Ｈ０４Ｌ　11/00	３２０	

審査請求　未請求　請求項の数52　ＯＬ　（全 22 頁）

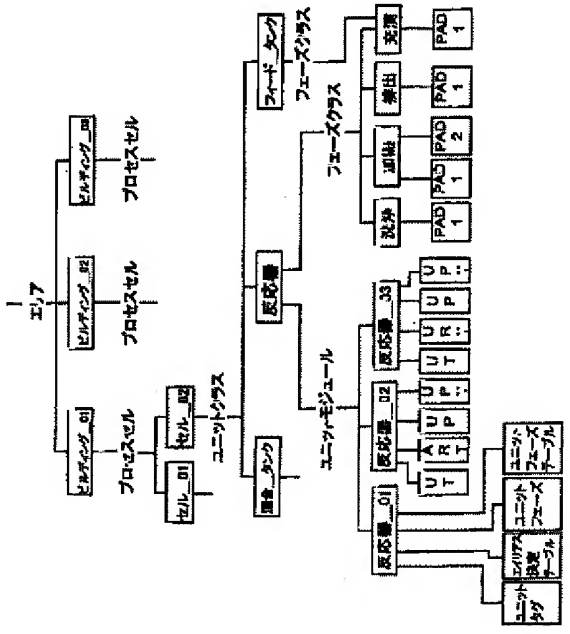
(21)出願番号　特願2000－69117(P2000－69117)
(22)出願日　平成12年３月13日(2000.3.13)
(31)優先権主張番号　０９／２６７４３１
(32)優先日　平成11年３月12日(1999.3.12)
(33)優先権主張国　米国（ＵＳ）

(71)出願人　594120847
フィッシャー－ローズマウント　システムズ，　インコーポレイテッド
アメリカ合衆国　78754　テキサス　オースティン　キャメロン　ロード　8301
(72)発明者　アーウィン，　ウィリアム　ジー．
アメリカ合衆国　78730　テキサス　オースティン　イーグル　トレース　トレイル　5116
(74)代理人　100065868
弁理士　角田　嘉宏　（外４名）

最終頁に続く

(54)【発明の名称】　プロセス制御ルーチンに於ける間接参照

(57)【要約】
【課題】　現在コントローラによって実行されている制御ルーチンが変更されても、そのプロセスの中止を生ずることのないプロセス制御システムを提供する。
【解決手段】　それぞれが一又はそれ以上のユニットモジュールを含む複数のユニットクラスを有するプロセスの制御に於いて使用するためのプロセス制御システムに、コントローラと、メモリと、エイリアス名を使用する一般的な制御ルーチンと、そのエイリアス名のエイリアス定義をそれぞれ有し、ユニットクラスの一つのユニットモジュールのためのエイリアス決定テーブルとを設ける。一般的な制御ルーチンはメモリ内に格納され、ユニットモジュールの特定の一つの制御が必要とされたとき、コントローラは特定の一つのユニットモジュールのためのエイリアス決定テーブルを使用して、特定の一つのユニットモジュールの一般的な制御ルーチンのインスタンス化されたバージョンを生成し、特定の一つのユニットモジュールを前記一般的な制御ルーチンのインスタンス化されたバージョンを実行することにより制御を行う。



【特許請求の範囲】

【請求項１】　それぞれが一又はそれ以上のユニットモジュールを含む複数のユニットクラスを有するプロセスの制御に於いて使用するためのプロセス制御システムであって、該プロセス制御システムが、

コントローラと、

メモリと、

エイリアス名を使用する一般的な制御ルーチンと、

そのエイリアス名のためのエイリアス定義をそれぞれ有し、ユニットクラスの一つのユニットモジュールのそれぞれのためのエイリアス決定テーブルとを有し、前記一般的な制御ルーチンはメモリ内に格納され、前記ユニットモジュールの特定の一つの制御が必要とされたとき、前記コントローラは前記特定の一つのユニットモジュールのためのエイリアス決定テーブルを使用して、前記特定の一つのユニットモジュールの一般的な制御ルーチンのインスタンス化されたバージョンを生成し、前記特定の一つのユニットモジュールを前記一般的な制御ルーチンのインスタンス化されたバージョンを実行することにより制御するプロセス制御システム。

【請求項２】　請求項１記載のプロセス制御システムであって、前記一般的な制御ルーチンは、前記ユニットクラスの２又はそれ以上の選択されたユニットクラスに関連するユニットモジュールを制御するために適用されるように適合しており、前記選択されたユニットクラスのそれぞれに対するユニットモジュールのそれぞれについてのエイリアス決定テーブルは前記エイリアス名のためのエイリアス定義を有しているプロセス制御システム。

【請求項３】　請求項２記載のプロセス制御システムであって、前記コントローラに通信可能に接続された構成ワークステーションを更に含み、前記ワークステーションは、構成ルーチンを格納するワークステーションメモリと、前記構成ルーチンを実行するプロセッサとを備え、前記構成ルーチンは、前記選択されたユニットクラスのそれぞれに対するユニットモジュールのそれぞれについてのエイリアス決定テーブルがそのエイリアス名についての正しいエイリアス定義を含んでいるかどうかを決定するチェックルーチンを含んでいるプロセス制御システム。

【請求項４】　請求項２記載のプロセス制御システムであって、前記一般的な制御ルーチンがその一般的制御ルーチンのインスタンス化されたバージョンを生成するために使用され得る前記選択されたユニットクラスを示す一般的な制御ルーチンに関連する指標を更に有しているプロセス制御システム。

【請求項５】　請求項１記載のプロセス制御システムであって、ユニットモジュールからユニットモジュールへと変化するユニットモジュール特性を示す少なくとも一つのユニットクラスのユニットモジュールのそれぞれについての指標を更に含み、前記一般的制御ルーチンは前記ユニットモジュール特性に基づいて異なるユニットモジュールを制御するのに２者択一的に使用されるべき少なくとも２つの制御アルゴリズムを含み、前記特定のユニットモジュールのための一般的な制御ルーチンののインスタンス化されたバージョンを生成したときに、前記コントローラは、前記少なくとも２つの制御アルゴリズムの一つを選択するために、前記ユニットモジュールの特定の一つのための前記指標を使用する、プロセス制御システム。

【請求項６】　請求項１記載のプロセス制御システムであって、前記ユニットモジュールの一つのためのエイリアス決定テーブルは、前記ユニットモジュールの一つのための前記一般的な制御ルーチンのインスタンス化されたバージョンを実行するときに、前記コントローラが前記エイリアス名を無視するようにさせるエイリアス名のための無視エイリアス定義を含んでいるプロセス制御システム。

【請求項７】　請求項１記載のプロセス制御システムであって、前記一般的制御ルーチンは動的参照パラメータを含み、その値は前記特定のユニットモジュールのための一般的制御ルーチンのインスタンス化されたバージョンの生成後に割り当てられる、プロセス制御システム。

【請求項８】　請求項７記載のプロセス制御システムであって、前記動的参照パラメータは複数の属性を有し、その属性の一つは、前記動的参照パラメータが示すフィールドを特定するフィールド値を保持している参照属性である、プロセス制御システム。

【請求項９】　請求項８記載のプロセス制御システムであって、前記属性の他のものは、前記参照属性の前記フィールド値が正しいフィールドであるかどうかの指標を提供する接続属性である、プロセス制御システム。

【請求項１０】　請求項８記載のプロセス制御システムであって、前記属性の他のものは、前記参照属性によって特定されるフィールドの読み取り又は書き込みを行う読み取り／書き込み属性である、プロセス制御システム。

【請求項１１】　請求項８記載のプロセス制御システムであって、前記属性の他のものは、前記参照属性によって特定されるフィールドに関連する状態を読み取る状態属性である、プロセス制御システム。

【請求項１２】　請求項１１記載のプロセス制御システムであって、前記状態属性は、参照属性によって特定されたフェーズへの先の書き込みの成功又は失敗を示す書き込み状態を読み取る、プロセス制御システム。

【請求項１３】　それぞれが一又はそれ以上のユニットモジュールを含む複数のユニットクラスを有するプロセスの制御に於いて使用するためのプロセス制御システムであって、該プロセス制御システムが、

コントローラと、

メモリと、

エイリアス名を使用し、前記ユニットクラスの少なくとも２つの選択されたそれに関連する一又はそれ以上のユニットモジュールに適用されるように適合した一般的な制御ルーチンと、
そのエイリアス名のエイリアス定義をそれぞれ有し、前記選択されたユニットクラスのそれぞれのユニットモジュールのそれぞれのためのエイリアス決定テーブルと、を有し、前記一般的な制御ルーチンは、メモリ内に格納され、前記特定の一つのユニットモジュールのためのエイリアス決定テーブルを使用して、前記ユニットモジュールの特定の一つのための一般的な制御ルーチンのインスタンス化されたバージョンを生成するために使用されるプロセス制御システム。

【請求項１４】　請求項１３記載のプロセス制御システムであって、前記コントローラに通信可能に接続された構成ワークステーションを更に含み、前記ワークステーションは、構成ルーチンを格納するワークステーションメモリと、前記構成ルーチンを実行するプロセッサとを備え、前記構成ルーチンは、前記選択されたユニットクラスのそれぞれに対するユニットモジュールのそれぞれについてのエイリアス決定テーブルがそのエイリアス名についての正しいエイリアス定義を含んでいるかどうかを決定するチェックルーチンを含んでいるプロセス制御システム。

【請求項１５】　請求項１３記載のプロセス制御システムであって、一般的な制御ルーチンがその一般的制御ルーチンのインスタンス化されたバージョンを生成するために使用され得る前記選択されたユニットクラスを示す一般的な制御ルーチンに関連する指標を更に有しているプロセス制御システム。

【請求項１６】　それぞれが一又はそれ以上のユニットモジュールを含む複数のユニットクラスを有するプロセスの制御に於いて使用するためのプロセス制御システムであって、該プロセス制御システムが、
コントローラと、
メモリと、
ユニットモジュールからユニットモジュールへと変化するユニットモジュール特性を示す、ユニットクラスの少なくとも一つのユニットモジュールのそれぞれについての指標と、
異なるユニットモジュールのユニットモジュール特性に基づいて異なるユニットモジュールを制御するのに２者択一的に使用される少なくとも２つの制御アルゴリズム含む一般的な制御ルーチンと、
を有し、前記コントローラは、前記ユニットモジュールの特定の一つのための一般的な制御ルーチンのインスタンス化されたバージョンを生成するために、前記少なくとも２つの制御アルゴリズムの一つを選択する前記ユニットモジュールの特定の一つのための前記指標を使用する、プロセス制御システム。

【請求項１７】　プロセスの制御に於いて使用するためのプロセス制御システムであって、
コントローラと、
メモリと、
前記プロセスの少なくとも一部分を制御するのに使用される制御ルーチンとを有し、前記制御ルーチンはメモリに格納され、前記コントローラは前記制御ルーチンの実行可能なバージョンを生成し、前記制御ルーチンの前記実行可能なバージョンを実行することにより、前記プロセスの前記一部分の制御を行い、
前記制御ルーチンは、動的参照パラメータが指しているフィールドを特定するフィールド値を保持し、前記制御ルーチンの実行可能なバージョンの生成後に割り当てられ得る参照属性を含む複数の属性を有する動的参照パラメータを含む、プロセス制御システム。

【請求項１８】　請求項１７記載のプロセス制御システムであって、前記複数の属性の他のものは、前記参照属性の前記フィールド値が正しい参照であるかどうかの指標を提供する接続属性である、プロセス制御システム。

【請求項１９】　請求項１７記載のプロセス制御システムであって、前記属性の他のものは、前記参照属性によって特定されるフィールドの読み取り又は書き込みを行う読み取り／書き込み属性である、プロセス制御システム。

【請求項２０】　請求項１９記載のプロセス制御システムであって、前記読み取り／書き込み属性は、文字列値として読み取り又は書き込みを行うロセス制御システム。

【請求項２１】　請求項１９記載のプロセス制御システムであって、前記読み取り／書き込み属性は、数値として読み取り又は書き込みを行うロセス制御システム。

【請求項２２】　請求項１９記載のプロセス制御システムであって、前記読み取り／書き込み属性は、とブール値して読み取り又は書き込みを行うロセス制御システム。

【請求項２３】　請求項１９記載のプロセス制御システムであって、前記読み取り／書き込み属性は、配列値として読み取り又は書き込みを行うロセス制御システム。

【請求項２４】　請求項１７記載のプロセス制御システムであって、前記属性の他のものは、前記参照属性によって特定されるフィールドに関連する状態を読み取る状態属性である、プロセス制御システム。

【請求項２５】　請求項２４記載のプロセス制御システムであって、前記属性の他のものは、前記参照属性によって特定されるフィールドへの先の書き込みの成功又は失敗を示す書き込み状態を読み取る、プロセス制御システム。

【請求項２６】　請求項１７記載のプロセス制御システムであって、前記複数の属性の第２のそれは、前記参照属性の前記フィールド値が正しいフィールドであるかど

うかの指標を提供する接続属性であり、前記複数の属性の第３のそれは、前記参照属性によって特定されているフィールドの読み取り又は書き込みを行う読み取り／書き込み属性である、プロセス制御システム。

【請求項２７】　請求項２６記載のプロセス制御システムであって、前記複数の属性の第４のそれは、前記参照属性によって特定されるフィールドに関連する状態を読み取る状態属性である、プロセス制御システム。

【請求項２８】　それぞれ一又はそれ以上のユニットモジュールを含んだ複数のユニットクラスを有するプロセスを制御するコントローラを備えたプロセス制御システムに於いて使用されるソフトウエア制御コンポーネントであって、該ソフトウエア制御コンポーネントが、

コンピュータ読み取り可能なメモリと、

前記コンピュータ読み取り可能なメモリに格納され、エイリアス名を使用する一般的制御ルーチンと、

それぞれエイリアス名のためのエイリアス定義を有する、前記ユニットクラスの一つのユニットモジュールのそれぞれに対するエイリアス決定テーブルとを有し、前記一般的制御ルーチンはコントローラによって実行されるように適合し、これにより、前記ユニットモジュールの特定の一つの制御が必要とされたとき、前記一般的な制御ルーチンは前記コントローラによって使用されて、前記特定のユニットモジュールのための前記エイリアス決定テーブルを使用して前記特定のユニットモジュールの一般的な制御ルーチンのインスタンス化されたバージョンを生成する、ソフトウエア制御コンポーネント。

【請求項２９】　請求項２８記載のソフトウエア制御コンポーネントであって、前記一般的な制御ルーチンは、前記ユニットクラスの２又はそれ以上の選択されたユニットクラスに関連する一又はそれ以上のユニットモジュールを制御するのために適用されるように適合しており、前記選択されたユニットクラスのそれぞれに対するユニットモジュールのそれぞれについてのエイリアス決定テーブルは前記エイリアス名のためのエイリアス定義を有しているソフトウエア制御コンポーネント。

【請求項３０】　請求項２９記載のソフトウエア制御コンポーネントであって、一般的な制御ルーチンがその一般的制御ルーチンのインスタンス化されたバージョンを生成するために使用され得る前記選択されたユニットクラスを示す一般的な制御ルーチンに関連する指標を更に有しているソフトウエア制御コンポーネント。

【請求項３１】　請求項２８記載のソフトウエア制御コンポーネントであって、前記ユニットモジュールの一つのためのエイリアス決定テーブルは、前記ユニットモジュールの一つのための前記一般的な制御ルーチンのインスタンス化されたバージョンを実行するときに、前記コントローラが前記エイリアス名を無視するようにさせるエイリアス名のための無視エイリアス定義を含んでいるプロセス制御システム。

【請求項３２】　請求項２８記載のソフトウエア制御コンポーネントであって、ユニットモジュールからユニットモジュールへと変化するユニットモジュール特性を示す、前記ユニットクラスの少なくとも一つのユニットモジュールのそれぞれについての指標を更に有し、前記一般的な制御ルーチンは、前記ユニットモジュール特性に基づいて異なるユニットモジュールを制御するのに２者択一的に使用されるように適合した少なくとも２つの制御アルゴリズム含み、これにより、前記コントローラは、前記特定の前記ユニットモジュールのための一般的な制御ルーチンのインスタンス化されたバージョンを生成したとき、前記少なくとも２つの制御アルゴリズムの一つを選択するように、前記ユニットモジュールの特定の一つのための前記指標を使用する、ソフトウエア制御コンポーネント。

【請求項３３】　請求項２８記載のソフトウエア制御コンポーネントであって、前記一般的制御ルーチンは動的参照パラメータを含み、その値は前記特定のユニットモジュールのための一般的制御ルーチンのインスタンス化されたバージョンの生成後に割り当てられる、ソフトウエア制御コンポーネント。

【請求項３４】　請求項３３記載のソフトウエア制御コンポーネントであって、前記動的参照パラメータは複数の属性を有し、その属性の一つは、前記動的参照パラメータが示すフィールドを特定するフィールド値を保持している参照属性である、ソフトウエア制御コンポーネント。

【請求項３５】　請求項３４記載のソフトウエア制御コンポーネントであって、前記複数の属性の他のものは、前記参照属性の前記フィールド値が正しいフィールドであるかどうかの指標を提供する接続属性である、ソフトウエア制御コンポーネント。

【請求項３６】　請求項３４記載のソフトウエア制御コンポーネントであって、前記属性の他のものは、前記参照属性によって特定されるフィールドの読み取り又は書き込みを行う読み取り／書き込み属性である、ソフトウエア制御コンポーネント。

【請求項３７】　請求項３４記載のソフトウエア制御コンポーネントであって、前記複数の属性の他のものは、前記参照属性によって特定されるフィールドに関連する状態を読み取る状態属性である、ソフトウエア制御コンポーネント。

【請求項３８】　それぞれ一又はそれ以上のユニットモジュールを含んだ複数のユニットクラスを有するプロセスを制御するコントローラを備えたプロセス制御システムに於いて使用されるソフトウエア制御コンポーネントであって、該ソフトウエア制御コンポーネントが、

コンピュータ読み取り可能なメモリと、

前記コンピュータ読み取り可能なメモリに格納され、エイリアス名を使用し、前記ユニットクラスの選択された

少なくとも２つのクラスに関連する一又はそれ以上のユニットモジュールを制御するために適用されるように適合した一般的制御ルーチンと、
それぞれエイリアス名のためのエイリアス定義を有する、前記選択されたユニットクラスのユニットモジュールのそれぞれに対するエイリアス決定テーブルとを有し、前記一般的制御ルーチンは、前記ユニットモジュールの特定の一つのためのエイリアス決定テーブルを使用して、前記ユニットモジュールの特定の一つのための一般的な制御ルーチンのインスタンス化されたバージョンを生成するために、コントローラによって使用されるように適合している、ソフトウエア構成成分。
【請求項３９】　請求項３８記載のソフトウエア制御コンポーネントであって、前記選択されたユニットクラスのそれぞれに対するユニットモジュールのそれぞれについてのエイリアス決定テーブルがそのエイリアス名についての正しいエイリアス定義を含んでいるかどうかを決定するチェックルーチンを有する構成ルーチンを含んでいるソフトウエア制御コンポーネント。
【請求項４０】　請求項３８記載のソフトウエア制御コンポーネントであって、一般的な制御ルーチンがその一般的制御ルーチンのインスタンス化されたバージョンを生成するために使用され得る前記選択されたユニットクラスを示す一般的な制御ルーチンに関連する指標を更に有しているソフトウエア制御コンポーネント。
【請求項４１】　それぞれ一又はそれ以上のユニットモジュールを含んだ複数のユニットクラスを有するプロセスを制御するコントローラを備えたプロセス制御システムに於いて使用されるソフトウエア制御コンポーネントであって、該ソフトウエア制御コンポーネントが、
コンピュータ読み取り可能なメモリと、
ユニットモジュールからユニットモジュールへと変化するユニットモジュール特性を示す、ユニットクラスの少なくとも一つのユニットモジュールのそれぞれについての指標と、
前記コンピュータ読み取り可能なメモリに格納され、異なるユニットモジュールのユニットモジュール特性に基づいて異なるユニットモジュールを制御するのに２者択一的に使用される２つの制御アルゴリズム含む一般的な制御ルーチンと、
を有し、前記指標は、前記コントローラが特定の前記ユニットモジュールの一つのための一般的な制御ルーチンのインスタンス化されたバージョンを生成したとき、前記ユニットモジュールの特定の一つが前記２つの制御アルゴリズムの一つを選択するように、前記コントローラによって使用される、ソフトウエア構成成分。
【請求項４２】　プロセスを制御するコントローラを備えたプロセス制御システムに於いて使用されるソフトウエア制御コンポーネントであって、該ソフトウエア制御コンポーネントが、
コンピュータ読み取り可能なメモリと、
前記コンピュータ読み取り可能なメモリに格納され、前記プロセスの少なくとも一部分を制御するために前記コントローラよって使用されるように適合した制御ルーチンとを有し、前記制御ルーチンは、動的参照パラメータが指すフィールドを特定するフィールド値を保持し、前記制御ルーチンの実行可能なバージョンの生成後に割り当てられ得る参照属性を含む複数の属性を有する動的参照パラメータを含む、ソフトウエア制御コンポーネント。
【請求項４３】　請求項４２記載のソフトウエア制御コンポーネントであって、前記複数の属性の他のものは、前記参照属性の前記フィールド値が正しいフィールドであるかどうかの指標を提供する接続属性である、ソフトウエア制御コンポーネント。
【請求項４４】　請求項４２記載のソフトウエア制御コンポーネントであって、前記属性の他のものは、前記参照属性によって特定されるフィールドの読み取り又は書き込みを行う読み取り／書き込み属性である、ソフトウエア制御コンポーネント。
【請求項４５】　請求項４４記載のソフトウエア制御コンポーネントであって、前記読み取り／書き込み属性は、文字列値として読み取り又は書き込みを行うソフトウエア制御コンポーネント。
【請求項４６】　請求項４４記載のソフトウエア制御コンポーネントであって、前記読み取り／書き込み属性は、数値として読み取り又は書き込みを行うソフトウエア制御コンポーネント。
【請求項４７】　請求項４４記載のソフトウエア制御コンポーネントであって、前記読み取り／書き込み属性は、ブール値として読み取り又は書き込みを行うソフトウエア制御コンポーネント。
【請求項４８】　請求項４４記載のソフトウエア制御コンポーネントであって、前記読み取り／書き込み属性は、配列値として読み取り又は書き込みを行うソフトウエア制御コンポーネント。
【請求項４９】　請求項４２記載のソフトウエア制御コンポーネントであって、前記複数の属性の他のものは、前記参照属性によって特定されるフィールドに関連する状態を読み取る状態属性である、ソフトウエア制御コンポーネント。
【請求項５０】　請求項４９記載のソフトウエア制御コンポーネントであって、前記参照属性は、前記参照属性によって特定されるフィールドへの先の書き込みの成功又は失敗を示す書き込み状態を読み取る、ソフトウエア制御コンポーネント。
【請求項５１】　請求項４２記載のソフトウエア制御コンポーネントであって、前記複数の属性の第２のそれは、前記参照属性の前記フィールド値が正しいフィールドであるかどうかの指標を提供する接続属性であり、前

記複数の属性の第３のそれは、前記参照属性によって特定されているフィールドの読み取り又は書き込みを行う前記読み取り／書き込み属性である、ソフトウエア制御コンポーネント。

【請求項５２】　請求項５１記載のソフトウエア制御コンポーネントであって、前記複数の属性の第４のそれは、前記参照属性によって特定されるフィールドに関連する状態を読み取る状態属性である、ソフトウエア制御コンポーネント。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、概略的にはプロセス制御ネットワークに関し、より詳細には、改良されたプロセス制御を可能にするためのプロセス制御ルーチンに於ける間接参照に関する。

【０００２】

【従来の技術】化学、石油又は他のプロセスで使用されるプロセス制御ネットワークは、一般的に、例えばバルブポジショナ、スイッチ、（温度、圧力及び流速センサのような）センサ等の一又はそれ以上のフィールドデバイスに通信可能に結合された集中処理コントローラを含んでいる。これらのフィールドデバイスは、（バルブの開閉などの）プロセスに於ける物理的な制御機能を果たし、プロセスの運転の制御に使用するためにプロセスの測定を行い、又はプロセス内の他の所望の機能を果たし得る。プロセスコントローラは、歴史的には、例えばフィールドデバイスへ及びフィールドデバイスからの４－２０ｍＡ（ミリアンペア）の信号を運ぶ一又はそれ以上のアナログ信号ライン又はバスを介してフィールドデバイスに接続されている。しかし、近年になって、プロセス制御産業は多くのスタンダードでオープンなデジタル又はデジタルとアナログとを結合した通信プロトコル、例えばFOUNDATION、FIELDBUS（以後「Fieldbus」という。）、HART、PROFIBUS、WORLDFIP、Device-Net及びCANのようなコントローラとフィールドデバイスとの間の通信を行うのに使用されるプロトコルが開発されている。一般的に言えば、プロセスコントローラは、一又はそれ以上のフィールドデバイス及び／又はフィールドデバイスに関連する他の情報によって作成される測定値を表す信号を受け取り、この情報を典型的に複雑な制御ルーチンを実行するのに使用し、及びフィールドデバイスに信号ライン又はバスを介して送られる制御信号を生成し、これによってプロセスの運転の制御を行う。

【０００３】バッチプロセスで使用されるような特定のタイプのプロセス制御ネットワークは、プロセス内で本質的に同じ機能を行う同一又は類似の装置を有するように設計され複製された装置の複数のセットを典型的に含んでいる。従って、例えば、クッキーの製造プラントは、混合装置の複数のセット、ベーキング装置の複数のセット及び包装装置の複数のセットを有し、個々のミキサーの全ては並行して動作することができ、ベーキング装置のいずれかと、及び包装装置のいずれかと、直列に動作するように接続することができる。このようなシステムでは、複製された特定のセットのいずれの動作をも制御し、これによってコントローラ内で生成され格納されるべき多くの制御ルーチンを低減させるように、同じ制御アルゴリズム及びルーチンを使用し得ることが好ましい。しかしながら、これらの制御アルゴリズムは、実行されたときにそのときに使用されている特定のユニットの装置を特定するように書かれていなければならない。

【０００４】幾つかの従来技術のシステムでは、エイリアスの名称（即ち、特定されない変数）を使用して複製ユニットから複製ユニットへ変化する特定の装置を指定するための制御ルーチンは、ワークステーションに生成される。システムコントローラが特定のユニット上で生成された制御ルーチンを実行することを可能にするために、一般化されたプログラムは、特定の装置のために生成されたエイリアス決定テーブルを使用してインスタンス化される。このようなエイリアス決定テーブルは、一般化された制御ルーチンで使用される各エイリアス名のための定義を含み、制御ルーチンに於けるエイリアス名のための特定のユニットのエイリアス決定テーブルに於ける値を置換することにより、一般化された制御ルーチンの実行可能なインスタンスを生成するのに使用される。このインスタンス化された制御ルーチンは、次に、コントローラにダウンロードされ格納され、その後、特定の装置上で制御動作（又はフェーズ）を実行する実行時間の間に使用される。しかしながら、このシステムを使用すると、コントローラは、複製されたユニットの一つの異なるそれぞれに対するインスタンス化された（決定された）制御ルーチン分離しなければならなず、これにより、特にもし、コントローラが非常に多くの同様なユニットを制御するのに使用されるなら、そして各ユニット上で多くの異なる動作とフェーズを実行するために使用されているなら（別々のインスタンス化された制御ルーチンが各ユニットの各フェーズに必要とされるため）、コントローラ内に多くのメモリ空間が必要となる。

【０００５】

【発明が解決しようとする課題】他の従来技術のシステムでは、生成された制御ルーチンは、コントローラ内に格納され、それが適用されるべき複製されたユニットの何れかの上に於いてプログラムされた動作又はフェーズを実行するための実行時間の間に使用される。この場合には、エイリアス名は、制御されている特定のユニットのためのエイリアス決定テーブルを使用して、実行時間中に大急ぎで決定される。しかしながら、この構成では、もし現在コントローラによって実行されている一般化された制御ルーチンに変更が加えられると、そのルー

チンの実行は中止してそのコントローラにダウンロードされるべき新たに生成された制御ルーチンを実行可能としなければならない。その結果として、そのプロセスの中止された実行に関連する材料、時間などのロスが生ずる。

【０００６】更に、これらの公知のシステムの何れもが、複数の又は異なるクラスのユニット又は装置を超えて適用されるべきエイリアス名を有する単一の一般的なプロセス制御ルーチンを可能とはしていない。事実、これらの従来技術のシステムでは、フェーズのための制御ルーチンは、一つのユニットクラス、即ち、反応器、ミキサー等の一つの特定のタイプのハードウエアユニットについての使用に限定されている。結果として、例えば、反応容器を満たすために第１の一般的なプロセス制御ルーチンが生成され格納されなければならず、一方、ミキシングタンクを満たすために第２の一般的なプロセス制御ルーチンが生成され格納されなければならず、フィーダータンクを満たすために第３のそれが生成され格納されなければならず、結果的に異なるタイプのハードウエア上の本質的に同じ機能を果たすための多くの異なる一般的な制御ルーチンを生成することになる。

【０００７】同様に、これらの従来技術のシステムのいずれも、特定のタイプのハードウエアユニットの異なるモジュールに関連する装置間の相違を捕らえた一般的な制御ルーチンを可能としたものではない。例えば、もし第１の反応器ユニットがこれに関連する電気的な加熱エレメントを有し、第２の反応器ユニットがこれに関連するスチーム加熱エレメントを有しているなら、たとえ、加熱のみが必要なプロセスが不適切な加熱のタイプで実行されていても、電気的ヒーターとスチームヒーターとの制御に於ける相違を捕らえたこれらの反応器ユニットのそれぞれを加熱するために、異なる一般的な制御ルーチンが生成されなければならない。この問題は、付加的なユニット又はコスト的な理由により（反応モジュールのような）モジュールが様々な時点でプロセス制御システムに追加されたとき、ハードウエアが改良されたときなどに生じ、新たに追加されたモジュールは、現存するモジュールと本質的に同じ機能を果たすように設計されているけれども、これに関連するわずかに異なる装置を有している。

【０００８】更にまた、これらの従来技術のシステムは、プロセスのいずれのフェーズについてもプロセス制御ルーチンの実行時間中に確認されるべきパラメータを特定する容易な方法を有していない。事実、間接参照を有する従来技術のシステムの殆どに於いては、実行時間の前であるプロセス制御ルーチンが構成されるとき又は機械可読コードに変換されるときに、エイリアス決定テーブルがエイリアス名を決定するのに使用されている。変数を実行時間の間に変更又は特定可能とするために、制御プログラムが配列中のアドレスの一つを参照する命令に到達したときにプログラムが特定されたアドレスの内容によって参照されるデバイス又はポジションに到達し得るように、実行時間の間に参照又はポインタがそこに置かれ得るようなアドレス配列等のアドレススキームを、ある従来技術のシステムでは提供している。しかしながら、参照アドレスの内容が実行時間の間制御ルーチン内の正確なデバイス又は適切な場所を指しているかどうかを教示する方法はない。もしポインタが正確でなければ、プログラムは停止し継続することはできず、生産は停止することになる。更に、このスキームは複雑で正確に使用することが困難である。なぜなら、ポインタを保持しているアドレス配列の詳細な知識と、その時点で制御ルーチンによってどの配列のアドレスが使用されているかについての知識とが必要となるからである。従って、設計者は多くの労力を必要とし、ユーザは、実行時間の間に停止することがないように、正確な時間に正確なアドレスに正しいポインタが確実に格納されるようにすることが必要となる。

【０００９】

【課題を解決するための手段】プロセス制御システムは、エイリアス名及び／又は動的参照パラメータを使用する間接参照を可能とする一又はそれ以上のプロセス制御ルーチンを含んでいる。一般的なプロセス制御ルーチンは、エイリアス名を含むように書かれ、この一般的なプロセス制御ルーチンは、例えば、その中に複製された装置（複製されたユニット）を有するプロセスを制御するコントローラ内に格納される。特定のユニット上でプロセス制御機能を実行する前に、その機能を制御する一般的な制御ルーチンのインスタンスが生成され、そこではその一般的なルーチンに於けるエイリアス名が、特定のユニットのためのエイリアス名決定テーブルに定義されているパラメータによって置き換えられる。コントローラは、次に、そのユニットの運転の制御を行うために一般的なルーチンのインスタンス化されたバージョンを実行する。このことは、コントローラのメモリの必要量を低減させる。なぜなら、それは、コントローラが、常に全てのユニットに対する一般的ルーチンのインスタンス化されたバージョンを格納することに代えて、一般的なルーチンと現在実行されているルーチンのインスタンス化されたバージョンのみを格納することを可能とするからである。更に、このことは、そのインスタンスの間に変更されるべき一般的な制御ルーチンが停止するルーチンを実行することを引き起こすことなく実行されることを可能とする。

【００１０】所望なら、一般的なプログラムはそれに関連する多重アルゴリズムを有することができ、そこでは、異なるアルゴリズムのそれぞれは、たとえプロセス制御システム内の本質的に同じ機能を異なるユニットが実行する場合であっても、幾分異なるハードウエアを有する異なるユニットを制御するように設計されている。

一般的なプログラムのインスタンス化されたバージョンが生成されたとき、コントローラは、インスタンス化された制御ルーチンが生成されている特定のユニットのハードウエア構成を特定する格納された指標に基づいて、一般的ルーチンの複数のアルゴリズムのどれを使用すべきかを決定する。また、このシステムは、制御ルーチンが、多重クラス又はプロセス制御システム内の異なる機能のために使用されるハードウエアの異なるタイプのために生成され及び適用されることを可能とする。この場合には、一般的な制御ルーチンが適用されるべきユニットの異なるクラスに関連するエイリアス決定テーブルに各エイリアス名のためにエイリアス定義が存在することを保証し得る。このことは、プロセス制御システム内の異なる目的のために使用される異なるタイプの装置上で特定の機能を実行するのに、一つの一般的なプロセス制御ルーチンが書かれ及び使用されるので、書かれてコントローラに格納されるべき一般的なプロセス制御ルーチンをより少なくすることができる。

【００１１】更に、プロセス制御ルーチンは、インスタンス化された実行可能なプロセスが生成された後にフィールド、デバイス、パラメータなどが特定されるのを可能とする、即ち、実行時又は実行期間の間に動的に参照されるべきフィールドを可能とする、一又はそれ以上の動的参照パラメータを使用し得る。動的参照パラメータは、多重属性を有し、これには、例えば、ポインタ、参照されたデバイス、フィールド、パラメータなどへのパス又はタグ、参照属性によって特定されるフィールドへの実際の接続が行われ得るかどうか、即ち、参照属性がプロセス制御システム構成内の正しいフィールドを定義しているかを示す接続属性が含まれる。また、動的参照パラメータは、文字列又は数値としての参照属性によって特定されるフィールドからの読み取り／及び／又はこれへの書き込みを可能とする属性を含んでいる。更にまた、動的参照パラメータは、例えば、そのフィールドの状態とそのフィールドに最後に書かれた状態とに関連する一又はそれ以上の状態値の読み取りを可能とする一又はそれ以上の属性を含んでいる。

【００１２】

【発明の実施の形態】図１を参照すれば、プロセス制御ネットワーク１０は、イーサネット（登録商標）接続１５を介して多くのワークステーション１４に接続されたプロセスコントローラ１２を有している。また、コントローラ１２は、（参照符号１６で概略的に示されている）プロセス内のデバイス又は装置に、通信ラインのセット又はバス１８を介して接続されている。ここで、例示として、コントローラ１２は、フィッシャーローズマウント社によって販売されているＤｅｌｔａＶ（登録商標）コントローラであり、これは、一又はそれ以上のプロセス制御ルーチンを実行しそれによってプロセス１６の所望の制御を実行するために、フィールドデバイスのような制御エレメント及びプロセス１６全体に分散したフィールドデバイス内の機能ブロックと通信可能になっている。ワークステーション１４（これは、例えば、パーソナルコンピュータである）は、プロセス制御ルーチンなどをダウンロードするようにコントローラ１２と通信しそしてプロセス１６の運転の間にプロセス１６に関する情報を受け取り及び表示するために、コントローラ１２によって実行されるべきプロセス制御ルーチンを設計するように、一又はそれ以上のエンジニア又はユーザによって使用され得る。ワークステーション１４のそれぞれは、構成設計アプリケーション等のアプリケーションを格納するため、及びプロセス１６の構成に関する構成データのようなデータを格納するためのメモリ２０を有している。ワークステーション１４のそれぞれはまた、ユーザがプロセス制御ルーチンを設計しこれらのプロセス制御ルーチンをコントローラ１２にダウンロードするのを可能とするアプリケーションを実行するためのプロセッサ２１を有している。同様に、コントローラ１２は、プロセス１６の制御に使用されるべき構成データ及びプロセス制御ルーチンを格納するためのメモリ２２と、プロセス制御戦略を実行するためにプロセス制御ルーチンを実行するためのプロセッサ２４とを有している。もしコントローラ１２がＤｅｌｔａＶコントローラであるなら、コントローラ１２内のプロセス制御ルーチンのグラフィカルな描写を、プロセス制御ルーチン内の制御エレメントを例証するワークステーション１４の一つを介して、これらの制御エレメントがプロセス１６の制御を提供するように構成されている方法で、ユーザに提供する。

【００１３】図１に示されているプロセス制御ネットワーク１０に於いては、コントローラ１２は、反応器＿０１、反応器＿０２及び反応器＿０３としてここに参照される同様に構成された反応器の３つのセットにバス１８を介して通信可能に接続されている。反応器＿０１は、反応容器１００と、反応容器１００に流体を提供する入力流体ラインを制御するように接続された入力バルブ１０１及び１０２と、出力流体ラインを介して反応容器１００から出る流体の流れを制御するように接続された出力バルブ１０３とを含んでいる。温度センサ、圧力センサ、流体レベルメーター等のセンサであり得るデバイス１０５は、反応容器１００に又はその近傍に配されている。同様に、反応器＿０２は、反応容器２００と、入力バルブ２０１及び２０２と、出力バルブ２０３と、デバイス２０５とを含み、反応器＿０３は、反応容器３００と、入力バルブ３０１及び３０２と、出力バルブ３０３と、デバイス３０５とを含んでいる。図１に示すように、コントローラ１２は、反応器ユニットに関連する一又はそれ以上の動作を遂行するために、これらの要素の動作を制御するように、バス１８を介してバルブ１０１から１０３、バルブ２０１から２０３及びバルブ３０１

から３０３、並びにデバイス１０５，２０５及び３０５に接続されている。このような動作には、例えば、反応容器の充満、反応容器内の材料の加熱、反応容器の排出、反応容器の洗浄などが含まれる。

【００１４】図１に示されているバルブ、センサ及び他の装置は、例えばフィールドバスデバイス、規格４－２０ｍａのデバイス、ＨＡＲＴデバイス等の所望の種類及びタイプの装置であり得、フィールドバスプロトコル、ＨＡＲＴプロトコル、規格４－２０ｍａのアナログプロトコル等の所望のプロトコル使用して、コントローラ１２と通信可能である。更に、本発明の原理に従って、他のタイプのデバイスが、コントローラ１２に接続され、これによって制御され得る。また、他のコントローラがコントローラ１２とワークステーション１４とにイーサネット通信ライン１５を介して、プロセス１６に関連する他のデバイス又はエリアを制御するために接続され得、このような付加的なコントローラは、図１に示されているコントローラ１２の動作に所望の方法で統制され得る。

【００１５】一般的に言えば、図１のプロセス制御システムは、例えばワークステーション１４又はコントローラ１２の一つがバッチ実行ルーチンを実行するバッチプロセスを実行するのに使用され、このバッチ実行ルーチンは、食品、薬等の製品を製造するために必要な一連の異なるステップ（一般的にフェーズと称される）を実行するための、（他の装置と同様に）一又はそれ以上の反応器ユニットの動作に向けられている高度なレベルの制御である。異なるフェーズを実行するために、バッチ実行ルーチンは、実行されるべきステップと、そのステップに関連する量と時間と、ステップの順序とを特定するレシピとして一般的に言及されるものを使用する。一つのレシピのステップは、例えば、反応容器を適当な材料又は成分で満たし、その材料を反応容器内で混合し、所定の時間内に所定の温度まで反応容器内の材料を加熱し、反応容器を空にし、次に、次のバッチ実行のために洗浄することを含んでいる。各ステップはバッチ実行のフェーズを定義し、コントローラ１２内のバッチ実行ルーチンがこれらのフェーズのそれぞれの一つのための異なる制御アルゴリズムを実行するであろう。もちろん、特定の材料、材料の量、加熱温度及び時間等は、異なるレシピごとに異なり、従って、これらのパラメータは、製造され又は生産されている製品と使用されているれしぴとに依存して、バッチ実行ごとに異なっている。当業者は、制御ルーチン及び構成が図１に示されている反応器で実行されるバッチの実行のためにここに記載され、制御ルーチンは、所望なら連続プロセスの実行を行うために他の所望のバッチプロセスの実行を行うための他の所望のデバイス制御するのに使用されることを理解するであろう。

【００１６】当業者が理解するように、バッチプロセスの同様のフェーズ又はステップが、同時に又は異なるときに図１の異なる反応器ユニットのそれぞれに於いて実行され得る。更に、図１の反応器ユニットは一般的に同じ数及びタイプの装置（即ち、それらは同じユニットクラスに属する）を含んでいるので、特定のフェーズのための同じ一般的フェーズ制御ルーチンは、この一般的フェーズ制御ルーチンが異なる反応器ユニットに関連する異なるハードウエア又は装置を制御するように改変されなければならないことを除いて、異なる反応器ユニットのそれぞれを制御するのに使用され得る。例えば、反応器＿０１（この反応器はここで満たされる）のための充満フェーズを実行するために、例えば、充満制御ルーチンが、流体レベルメーター１０５が反応容器１００が満たされたことを検知するまで一又はそれ以上の入力バルブ１０１又は１０２を所定の時間開く。しかしながら、この同じ制御ルーチンは、入力バルブの指定をバルブ１０２又は１０２に代えてバルブ２０１又は２０２に単に変更することにより、そして流体レベルメーターを流体レベルメーター１０５に代えて流体レベルメーター２０５に変更することにより、反応器＿０２の充満フェーズを実行するように使用され得る。

【００１７】以前に知られているシステムでは、一般化された制御ルーチンが、反応器（反応器＿０１、反応器＿０２又は反応器＿０３）の何れかに於ける特定のフェーズを実行するために、エイリアス名（即ち、一般的な変数）を使用して、特定の反応器ユニット又はその反応器ユニットのための特定のバルブ若しくはセンサのような未だに特定されないパラメータを示すために生成されたであろう。このように、（充満ルーチンのような）一般化された制御ルーチンが、制御ルーチンの生成に際して特定の反応容器を満たすために開けられる特定のバルブを特定することに代えて、図１の反応器ユニットの何れかの充満フェーズの間に使用されるように生成され、一般化された制御ルーチンは、使用される実際のバルブのためのエイリアスとして「入力＿バルブ」を単に特定する。これらのシステムのために、エイリアス決定テーブルが一般化された制御ルーチンが適用されるべきユニットのそれぞれのために生成される。例えば、反応器＿０１のためのエイリアス決定テーブルは、エイリアス「入力＿バルブ」がバルブ１０１であることを特定し、反応器＿０２のためのエイリアス決定テーブルは、エイリアス「入力＿バルブ」がバルブ２０１であることを特定する等である。

【００１８】上記に示したように、ある従来技術システムでは、これらのエイリアス名を使用する一般化されたプログラムはワークステーションに生成され、このプログラムのインスタンス化されたバージョンが、特定の反応器ユニット（又は他のモジュール）のためのエイリアス決定テーブルを使用して、各反応器ユニット（又は他のモジュール）のために生成される。これらの実例制御

ルーチンは、次に、コントローラにダウンロードされその中に格納され、その後、実行時間の間に特定のユニットに於けるフェーズを実行するために使用され、これは多くのメモリを必要とする。他の従来技術のシステムでは、一般化された制御ルーチンはコントローラ内に格納され、それが適用されるべきユニットの何れかのフェーズのプログラムされた動作を実行するために、実行時間の間に使用される。この場合には、実行時間の間の実行中に、制御されている特定のユニットのためのエイリアス決定テーブルを使用して、エイリアス名が決定される。この構成では、もし、コントローラによって現在実行されている一般化された制御ルーチンに変更が加えられたら、そのルーチンの実行は、そのコントローラにダウンロードされるべき新たな一般化された制御ルーチンを可能とするために停止されなければならない。

【００１９】更に、これらの公知のシステムの何れも、多重の若しくは異なるタイプのユニット又はハードウエアシステムを通じて適用されるエイリアス名を有する単一の一般的なプロセス制御ルーチンを可能とするものではない。事実、これらの従来技術のシステムでは、フェーズのための制御ルーチンは、一つのユニットクラス、即ち一つの特定のタイプのハードウエアユニットで、プロセス制御ネットワーク内の一つのタイプの機能に使用することに限定されている。結果として、最初の一般的なプロセス制御ルーチンは反応ユニットの充満のために生成され及び格納され、異なる一般的なプロセス制御ルーチンが混合タンクの充満のために生成され及び格納され、更に異なる一般的なプロセス制御ルーチンがフィードタンクの充満のために生成され及び格納され、結果的にハードウエアユニットの異なるタイプの本質的に同じ機能を実行するために、多くの異なる一般的制御ルーチンが生成されている。

【００２０】上述のように、これらの従来技術の何れも、特定のユニット又はユニットクラスの異なるモジュールに関連する装置間のわずかに異なる一般的制御ルーチンを賄うことを可能とするものではない。これに対して、もし、これらの異なるユニットが、例えばある場合には電気ヒーターであり他の場合にはスチームヒーターであるようなわずかに異なるハードウエアを有しているなら、異なるフェーズルーチンが書かれなければならず、同じユニットクラスの異なるモジュールで使用されなければならない。このことは、たとえこれらの異なるユニットがそのプロセス内で本質的に同じ機能を果たすとしても、プログラマーに、同じユニットクラスの異なるユニットのために異なるプロセス制御アルゴリズム又はフェーズクラスを使用することを要求することになる。更に、これらの従来技術システムには、プロセス制御ルーチンの実行時間の間に確認されるべきパラメータを特定するための簡便な方法がなく、そして、もしこのような動的参照が許容されるとしても、その動的参照が正しい値であるかどうかをその参照に基づいて命令を実行する前に知らせる方法が存在しない。

【００２１】ここに記載のプロセス制御プログラム又はルーチンに於ける間接参照を使用する手法又はシステムが、上記従来技術システムの問題のいくつかを解決する。ここに記述するシステムは、エイリアス名を使用して書かれ、最小限の格納スペースを使用してコントローラ内に未だ格納され、現在実行中のルーチンの何れかが停止されることなく変更される一般的なフェーズ制御ルーチンを可能とするような方法で実行されるべき一般的なプロセス制御ルーチン又はフェーズ制御ルーチンを可能とする。また、ここに記述のシステムは、異なる複数のユニットクラス又は異なるハードウエアのタイプに適応されるように生成されるべきフェーズ制御ルーチンを可能とし、動的参照、即ちフェーズ制御ルーチンの実行時間の間束縛される動的参照を検証し容易に使用することを含むフェーズ制御ルーチンを使用する。

【００２２】一般的に言えば、プロセス１６の動作がコントローラ１２内で管理され又は組織化される方法は、多くのオブジェクトの相互作用に基づいており、それぞれのオブジェクトは、属性を有し、それに関連する一又はそれ以上の方法を有し得る。オブジェクトのそれぞれは、それに関連する多くのサブオブジェクト（又はクラス）を有し、それぞれのサブオブジェクトは更にサブオブジェクトを有することができる。一般的な意味では、プロセス１６の全体に亘る制御計画は、その技術分野で知られ、そしてここでは更に詳細には記述しないが、オブジェクト指向プログラミングパラダイムを使用して構成される。図２は、図１のプロセス制御ネットワーク１０に関連する多くのオブジェクト間の相関関係を示すオブジェクト階層構造の例を表している。この階層構造は、例えば、プロセス制御ルーチンがワークステーション１４上に生成され、次に、ダウンロードされコントローラ１２内で実行されこのプロセス制御ルーチンが動作する状況を確認する方法を説明するのに使用される。

【００２３】図２のオブジェクトツリーは、特定のオブジェクトをボックスで囲み、一般的なオブジェクト（又はオブジェクトタイプ）のカテゴリーはボックスなしでツリー内のオブジェクトの上に示されている。図２に示すように、プロセス制御ネットワーク１０は、例えば、プラント内のビルディング又は他の地理的領域の表示である一又はそれ以上のエリアを含んでいる。図２のオブジェクトツリーでは、プロセス１６はビルディング＿０１、ビルディング＿０２及びビルディング＿０３と名付けられた３つのエリアオブジェクトを有している。各エリアオブジェクトはプロセスセルに分割され、それぞれのセルは、そのエリアで実行されているプロセスの異なる側面を示している。図２のビルディング＿０１エリアオブジェクトは、セル＿０１及びセル＿０２で表される２つのプロセスセルオブジェクトを含んでいるように示

されている。例えばセル_01はセル_02で使用される生成物の成分を作ることに関連している。各セルオブジェクトは、ゼロ又はそれ以上のユニットクラスを含み得、これはプロセスセルで使用されているハードウエアの異なるカテゴリ又はグルーピングを確認する。一般的に言えば、ユニットクラスは、複製された装置のセットの共通の構成を保持し、そして、より詳細には、もし同じでなければ非常に似たプロセス機器を有しそれぞれがもし同じでなければ非常に似た機能をプロセス内で実行するユニットの集合である命名されたオブジェクトである。ユニットクラスオブジェクトは、典型的にはそれが属するプロセス制御システム内のユニットのタイプを記述するように命名されている。図2は、混合_タンクユニットクラス、反応器ユニットクラス及びフィード_タンクユニットクラスを含んでいる。もちろん、殆どのプロセス制御ネットワークでは、ユニットクラスの多くの他のタイプ、例えばドライヤーユニット、フィードヘッダーユニット、及び他の個々の又はハードウエアの論理的グルーピングが提供され又は定義されるであろう。

【0024】図2の反応器ユニットクラスについて示されているように、各ユニットクラスオブジェクトは、それに関連するユニットモジュールオブジェクト及びフェーズクラスを有し得る。ユニットモジュールオブジェクトは、一般的には命名されたユニットクラス内の複製されたハードウエアの或るインスタンスを特定し、フェーズクラスは、一般的にそのユニットクラスに関連するユニットモジュールに適用され得るフェーズを特定する。より詳細には、ユニットモジュールオブジェクトは、単一のプロセスユニットのための全ての変数と（後に定義される）ユニットフェーズとを保持する命名されたオブジェクトであり、典型的には、特定のプロセス装置と同一に命名される。例えば、図2の反応器ユニットは、図1の反応器_01、反応器_02及び反応器_03にそれぞれ対応する反応器_01、反応器_02及び反応器_03のユニットモジュールを有している。混合_タンクユニットクラス及びフィード_タンクユニットクラスは、プロセス16内の特定のハードウエア又は装置に相当する特定のユニットモジュールを同様に有するであろう。しかしながら、簡単のために、混合_タンクユニットクラス又はフィード_タンクユニットクラスに関連する装置は、何れも図1には示されてはいない。

【0025】フェーズクラスは、同一のユニットクラスに、そして本発明に従えば複数の異なるユニットクラスに属する複数のユニット上で実行され得るフェーズのための共通の構成を保持している。本質的には、各フェーズクラスは、同一又は異なるユニットクラス内のユニットモジュールを制御するためにコントローラ12によって生成され使用される異なる制御ルーチン（又はフェーズ）である。典型的には、各フェーズクラスは、ユニットモジュール上で実行される動作を記述する動詞に従って命名される。例えば、図2に示すように、反応器ユニットは、図1の反応容器100，200又は300の何れか一つを満たすように使用される充満フェーズクラス、図1の反応容器100，200又は300の何れか一つを加熱するように使用される加熱フェーズクラス、図1の反応容器100，200又は300の何れか一つを排出するように使用される排出フェーズクラス、及び図1の反応容器100，200又は300の何れか一つを洗浄するように使用される洗浄フェーズクラスを有している。充満クラスは反応器ユニットクラスとフィード_タンクユニットの両方に関連し、本発明に従えば、フィード_タンクユニットモジュールのみならず、反応器ユニットモジュール上の充満機能をも果たすのに使用され得ことに注意すべきである。

【0026】フェーズクラスは、一般的には、バッチプロセスのためのレシピによって定義されるような、全体に亘るバッチプロセスで必要とされる幾つかの機能を実行するバッチ実行ルーチンによって呼び出されるサブルーチンと考えられている。フェーズクラスは、基本的にはバッチ実行ルーチン又は他のフェーズクラスからのフェーズクラスサブルーチンに提供される入力であるゼロ又はそれ以上の入力パラメータと、基本的にはバッチ実行ルーチン又は他のフェーズクラスに戻るフェーズクラスサブルーチンの出力であるゼロ又はそれ以上の出力パラメータと、そのフェーズクラスの動作とこのフェーズクラスが或る程度関連する他のフェーズクラスに関する情報とに関してユーザに表示されるゼロ又はそれ以上のフェーズメッセージと、フェーズ論理モジュール（PLM）又はこのフェーズクラスに基づくユニットフェーズで生成されるべきパラメータを生じさせるゼロ又はそれ以上のアルゴリズムパラメータとを有している。これらのアルゴリズムパラメータは、そのフェーズの実行中、一時格納位置又は変数として使用され、ユーザ又はバッチ実行ルーチンには必ずしも見えるものではない。重要なことは、フェーズクラスは一又はそれ以上のフェーズアルゴリズム定義（PAD）を含み、これは、一般的に言えば、そのフェーズを実行するのに使用される制御ルーチンである。また、フェーズクラスは、ゼロ、1，2又はそれ以上のユニットクラスへの関連のリストを有し、このリストは、このクラスと、従ってフェーズクラスのPADとが適用され得るユニットクラスを定義する。充満フェーズクラスリストの関連のリストは、反応器ユニット及びフィード_タンクユニットクラスの両方を含んでいる。

【0027】PADは、フェーズクラスのための要約又は一般的なフェーズ論理（アルゴリズム）を保持する命名されていないオブジェクトであり、絶対的なフェーズ論理状態マシンの何れか、連続するフローチャート（SFC）複合体、関数ブロック複合体、又はプロセス16の動作を制御するためにコントローラ12によって使用

されるどのような他の所望のプロセスプログラミングなど、どのような所望のタイプのプログラミング構造をも使用するように構成され得る。一実施例では、ＰＤＡはＳＦＣプログラミング技術を使用して定義され、そこでは多くのステップが互いに結合され、一つのステップ内でのアクションは遷移状態が真となるまで実行され、その時点で次の遷移状態が真になるまで実行される次のステップに制御が移行し、このような動作が続けられる。ＳＦＣプログラミングプロトコルは、プログラミング言語の規格IEC 848及びIEC 1131-3に基づいており、この技術分野でよく知られている。しかしながら、ＰＤＡはフェーズの動作を定義する他のどのような所望のタイプのプログラミング構造をも使用することができる。一般的に言えば、ＰＡＤは、バッチプロセスの動作の間コントローラ１２によって実行されるべきベース又は一般的制御ルーチンを提供する。

【００２８】一般的なＰＡＤがユニットクラス内の異なるユニットモジュールのどのような数にも適用することを可能とするために、ＰＡＤは、他のどのような所望の変数又はパラメータと同様に、エイリアス名即ち一般的に又はどのような特定のハードウエア又はハードウエア要素にも結合されていない未だ特定されていない名称を使用して、ユニットモジュールからユニットモジュールへ変化するどのような外部モジュール又はＩ／Ｏをも参照するように、（ユーザがワークステーション１４内の構成アプリケーションを使用して）構成される。結果として、ＰＡＤは複数のユニットモジュールに於いてそして複数のユニットクラスに於いてさえ適用しそれに於いて使用されるという意味に於いて一般的である。

【００２９】図２の各フェーズクラスは一つのＰＡＤオブジェクトを有し、例外的に加熱フェーズは２つのＰＡＤを有し、適用されるべき同じフェーズクラスを可能とし、以下の詳細に述べるように、これに関連する僅かに異なるタイプのハードウエア又は装置を有する制御ユニット（例えば反応器ユニット）を制御するのに使用される。

【００３０】図２のユニットモジュールを再び参照すれば、各ユニットモジュールオブジェクトはゼロ又はそれ以上のユニットタグ（ＵＴ）又は初期値を有するパラメータを含んでいる。これらのパラメータは、そのユニットモジュールに関連する装置の設定と構成パラメータに相当する。更に、各ユニットモジュールオブジェクトは、それに関連するアラーム、資源識別子、（人ーマシンインターフェイス画像のような）制御ディスプレイ、このユニットモジュールが必要とする資源のリスト、プロセスセル情報等を有している。重要なのは、各モジュールユニットオブジェクトは、それに関連するエイリアス決定テーブル（ＡＲＴ）、ゼロ又はそれ以上のユニットフェーズオブジェクト（ＵＰ）及びユニットフェーズテーブルを有していることである。エイリアス決定テーブルは、それが帰属するユニットモジュールのためのエイリアス名／インスタンス定義のペアのリストである。このテーブルのエイリアス名のリストは、ユニットクラスに対して定義された正しいエイリアス名に基づいており、従って、このユニットモジュールのユニットクラスに関連する全てのフェーズクラスで使用されるエイリアス名の全てを含んでいる。換言すれば、ユニットモジュールのエイリアス決定テーブルは、このユニットモジュール上で実行され得る各フェーズクラスで使用される各エイリアスのためのエイリアス定義を含んでいる。ユーザは、そのユニットクラスに関連するフェーズクで使用されるエイリアス名に基づいて、構成のときに各ユニット上で定義されるべきエイリアス名のためのインスタンス定義を構成する。

【００３１】幾つかのインスタンスでは、特定のエイリアス名を無視することを意味するエイリアスインスタンス定義のための特別の値を提供することが好ましい。この定義の効果は、少なくとも、このユニットモジュール上では、たとえこのユニットモジュールがそのエイリアス名を必要とせず又はサポートしない場合でも、このエイリアス名を使用するフェーズ論理はコントローラ１２にダウロードされるのを許容すべきであるということである。実行時間の実行は、エイリアスパラメータパスストリングを用いて特定され、抑制されるべき「無視」のエイリアスインスタンス定義を有するフェーズ論理のエイリアスパラメータに書き込みを試みることを引き起こし、抑制され又は値のない又はゼロ値又は他の幾つかの存在する値に戻すために「無視」として定義されるエイリアスパラメータの読込を試みることを引き起こし、あり得るなら問題が存在することをユーザに警告を発するようにアラームを引き起こす。所望なら、「無視」の定義がエイリアス定義の属性として格納され得、実行時間の間に、例えば、IF...THENの論理を使用してテストされ得る。

【００３２】各ユニットフェーズオブジェクトは、特定のユニットモジュールに関連し又はそのために生成されたフェーズクラスのインスタンスを表わす命名されたオブジェクトである。その構成システムに於いては（即ち、ワークステーション１４の一つに於いては）、ユニットフェーズオブジェクトは、独立に変更されダウンロードされ得るユニットモジュールの成分を表している。実行時間システムでは（即ち、コントローラ１２では）、ユニットフェーズオブジェクトは、ユニットモジュール上でコントローラ１２によって独立して操作（開始、停止、保持、中断等）され得るフェーズ論理を表し、異なるユニットモジュール上で同時にアクティブな他のユニットフェーズに潜在的に並行している。本質的には、ユニットフェーズオブジェクトは、そのユニットフェーズオブジェクトが属するユニットモジュールのためのエイリアス決定テーブルを使用して決定されたフェ

ーズクラスの一つのインスタンス化バージョンである。例えば、図１の一つの反応器＿０２のユニットフェーズオブジェクトは、反応器＿０２のモジュールユニットのためのエイリアス決定テーブルを使用してその中にエイリアス名を有する充満フェーズクラスのための一般的なＰＡＤを使用して生成されるであろう。従って、充満フェーズクラスのＰＡＤに於けるエイリアス入力＿バルブは、反応器＿０２のモジュールユニットのための充満ユニットフェーズオブジェクトに於ける図１のバルブ２０１又は２０２として定義され得る。コントローラ１２は、実際にユニットフェーズオブジェクト（即ち、実行時間の間のフェーズクラスのインスタンス化バージョン）を実行し、メモリ２２内にフェーズクラスの一般的バージョンを残す。

【００３３】ユニットモジュールのためのフェーズテーブルは、ユニットモジュール上で利用可能にされる全てのユニットフェーズについてのキー特性を保持する命名されていないオブジェクトである。フェーズテーブルは、ユニットモジュールのユニットクラスに帰属する全てのフェーズクラスのフェーズクラス名のリストを含んでいる。各フェーズクラスについて、ユーザは、ユニットフェーズ名（文字列）と、フェーズクラスで使用されている各エイリアス名のためのユニットモジュールのエイリアス名決定テーブルに於ける正しいエイリアスインスタンス定義が存在するか及び他の何れの必須フェーズクラスの検査チェックを通過したかの検査の識別子と、設計者又はユーザがフェーズクラスのためのユニットフェーズ論理をダウンロードするのを抑止することを可能とするダウンロードの識別子とを含むキー特性を見て／構成する。例えば、もし検査識別子が検査が行われていないことを特定し、又はもしダウンロード識別子がユーザによってＮＯにセットされているなら、ユニットフェーズはダウンロードされないであろう。また、フェーズテーブルはコントローラ１２にこのユニットフェーズが実行時間システムに於いて「永続」として扱われるべきであることを知らせる永続（(Yes/No)識別子を含んでいる。もしそうなら、ユニットフェーズは常に実例化されてコントローラメモリ資源の責務によって決して実行に失敗することはない。バッチ実行ルーチンによって要求される特性、資源識別子、必要な資源、コントローラ１２のための除算／乗算実行期間及びフェーズの実行時間の動作を制御する他の特性などの情報が、ユニットフェーズテーブルに提供される。

【００３４】図３は、図２に示されたオブジェクトの幾つかをより詳細に示したものであり、これらの間の相関関係をより良く示している。本発明の原理を示すために、２つのユニットクラス、即ち、反応器ユニットクラス５０及びフィード＿タンクユニットクラス５２が図３に示されている。反応器ユニットクラス５０はそのために示された一つのユニットモジュール５４、即ち反応器＿０１を有している。他のものも存在し得るが、それらは図３に単に示されていないだけである。ユニットモジュール５４は、反応器ユニットクラスに関連する反応器パラメータの幾つか、即ち、反応器＿０１の容量は３００で反応器＿０１は撹拌機を有していないことを定義している。同様に、２つのフェーズクラスは、充満フェーズクラス５６及び排出フェーズクラス５８を含む反応器ユニットクラスに関連している。充満フェーズクラス５６は、２つのエイリアス名即ち#INLET#VALVE# 及び #LEVEL#を使用して設計されたＰＡＤ（右側に図式的な形でＳＦＣとして表されている）を含んでいる。これらのエイリアス名は、充満フェーズクラス５６のＰＡＤに示されているボックス内で実際に使用され、さもなければＰＡＤの論理内のどこかで使用される。充満フェーズクラス５６はまた、目標＿レベルとして定義される入力と、最終＿レベルとして定義される出力とを含んでいる。エイリアス名は、数字の記号（＃）で区切られて示されており、他の識別子は、フェーズのインスタンス化に際して置き換えられなければならないエイリアス名を定義するのに使用され得る。同様に、排出フェーズクラス５８はＰＡＤを有し、これは右側に図式的な形で表されており、#OUTLET#VALVE# 及び #LEVEL#のエイリアス名を有し、速度として定義される入力と、最終＿レベルとして定義される出力と、実＿速度として定義されるアルゴリズムパラメータ（ＰＡＤによって使用される）とを有し、これらはＰＡＤの実行中に一時格納位置として使用される。

【００３５】充満フェーズクラス５６及び排出フェーズクラス５８は反応器ユニットクラス５０に関連しているので、反応器＿０１ユニットモジュール５４（これはまた反応器ユニットクラス５０にも関連している）はフェーズクラス５６及び５８の両方に使用されるエイリアス名を支持するように設計されるべきである。なぜなら、コントローラ１２は実行時間の間反応器＿０１ユニットモジュール５４のためにフェーズクラス５６及び５８のインスタンス化バージョンを生成しようとするからである。結果として、反応器＿０１ユニットモジュール５４のためのエイリアス決定テーブル６０は、各エイリアス名#INLET#VALVE#（充満フェーズクラス５６で使用される）と#LEVEL#（充満フェーズクラス５６と排出フェーズクラス５８との両方で使用される）と、#OUTLET#VALVE#（排出フェーズクラス５８で使用される）とを定義するように生成される。エイリアス決定テーブル６０は、#INLET#VALVE#と#LEVEL#エイリアスの正しい定義を含んでいるが、#OUTLET#VALVE#エイリアスの正しい定義を含んではいない。結果として、ワークステーション１４によって事項される構成ルーチンがそれぞれのフェーズクラスが制御スキームの中で正しく定義されたか否かを決定するとき、それは充満ユニットフェーズオブジェクトが反応器＿０１ユニットモジュール５４のために生成さ

れ得ることを決定する。なぜなら、充満フェーズクラスエイリアス名のそれぞれは、反応器＿０１ユニットモジュール５４のためのエイリアス決定テーブル６０で正しく決定されるからである。しかしながら、この構成ルーチンは、排出ユニットフェーズオブジェクトが反応器＿０１ユニットモジュール５４のために生成されないことを決定する。なぜなら、排出フェーズクラス５８によって使用されるエイリアス名の全てについて正しい定義を有していないからである。結果として、反応器＿０１ユニットモジュール５４のためのフェーズテーブル６２（これはワークステーション１４内の構成アプリケーションによって生成される）は、反応器＿０１ユニットモジュール５４のための充満フェーズが決定されコントローラ１２にダウンロードされるが、反応器＿０１ユニットモジュール５４のための排出フェーズは決定されず、従ってエイリアス決定テーブル６０の#OUTLET VALVE#エイリアスの正しくない定義に従って、これ１２にダウンロードされることはできない。

【００３６】図３に示されている充満フェーズクラス５６は、一般的には付加的な他のユニットクラス、即ちこれに関連するフィード＿タンク＿０６ユニットモジュール６４を有するように図３に示されているフィード＿タンク＿０６ユニットクラス５２に適用されるのに十分に定義されている。結果として、コントローラ１２にダウンロードされるように定義され又は適切にサポートされた充満フェーズクラスについては、（フィード＿タンクユニットクラス５２の他のユニットモジュールの全てと同様に）フィード＿タンク＿０６ユニットモジュールは、充満フェーズクラス５６、即ち#INLET#VALVE#及び#LEVEL#によって使用されるエイリアスに対する定義を提供するエイリアス決定テーブルを有していなければならない。このことが達成されるとき、充満フェーズクラス５６は反応器ユニットクラス５０又はフィード＿タンクユニットクラス５２の何れに於いても、どのようなユニットモジュール対してもユニットフェーズを生成するのに使用され得ることになる。

【００３７】構成の間、エンジニアなどのシステム設計者は、プロセス制御ネットワーク１０内のユニットモジュールのそれぞれに対してフェーズクラスとエイリアス決定テーブルとを構成するために、ワークステーション１４の一つで実行される構成プログラムを使用する。一般的に言えば、エンジニアは、何れか所望のプログラミング言語及び環境を使用して、そしてデバイス、モジュールパラメータ、関数ブロックなどの或る変数又はモジュールのためのエイリアス名を使用して、（図２の充満、排出、加熱及び洗浄のフェーズクラスのような）フェーズクラスのそれぞれに対するＰＡＤを定義する。これにより、ＰＡＤはそのフェーズクラスが帰属する何れかのユニットクラスに関連するユニットモジュールの何れをも制御するのに使用し又は適用され得ることになる。構成アプリケーションは、エンジニアが、これらのＰＡＤに何れか所望の方法で導入するのを可能とし、そして、例えばそれぞれのフェーズクラスに対するＰＡＤで使用されるエイリアス名のリストを含む必要な情報を特定するようにエンジニアに促し得る。

【００３８】本発明に従えば、エンジニアは、同一（又は異なる）ユニットクラスの異なるユニットモジュールに関連するハードウエア又は装置に於ける相違に起因して、どのような特定のフェーズクラスに対しても複数のＰＡＤを定義し得る。このように、例えば、反応器ユニットクラスに対する加熱フェーズクラスを生成するとき、エンジニアは、電気的加熱装置を使用して反応容器を加熱する第１のＰＡＤと、スティーム加熱装置を使用して反応容器を加熱する第２のＰＡＤとを提供し得る。これらのＰＡＤの異なる一つが、異なるユニットモジュールに対するユニットフェーズオブジェクトを生成するのに使用されるであろう。例えば、もし反応器＿０１ユニットモジュールがこれに関連する電気的加熱装置（例えば、図１のデバイス１０５は電気的加熱要素である）を有し、反応器＿０２がこれに関連する（例えば、図１のデバイス２０５はスティーム加熱要素である）を有しているなら、加熱フェーズクラスの第１のＰＡＤは反応器＿０１ユニットモジュールのための加熱フェーズクラスを生成するのに使用され、加熱フェーズクラスの第２のＰＡＤは反応器＿０２ユニットモジュールのための加熱フェーズクラスを生成するのに使用される。複数のＰＡＤを有するフェーズクラスのために特定のユニットフェーズを生成するときにワークステーション１４及びコントローラ１２がフェーズクラスのどのＰＡＤを使用するのかを決定することを可能とするために、複数のＰＡＤでフェーズクラスをサポートする各ユニットモジュールは、そのユニットモジュールが有している異なるＰＡＤ間の差異に関連する装置のタイプを確認する指標を含んでいるであろう。この識別子は、コントローラ１２が実行のたのにユニットフェーズを生成しているときにそれが適用可能である限り、何れの所望の方法によっても格納され得る。例えば、反応器＿０１ユニットモジュールは、この反応器が電気的加熱であることを特定する値にセットされている識別子パラメータを有し、一方、反応器＿０２ユニットモジュールは、この反応器がスティーム加熱であることを特定する値にセットされている識別子パラメータを有しているであろう。コントローラ１２は、ユニットモジュールのためのユニットフェーズを生成しているとき、識別子パラメータにアクセスし、ユニットフェーズを生成したとき、その識別子パラメータの値に基づいて、第１のＰＡＤ（電気的加熱）又は第２のＰＡＤ（スティーム加熱）を生成する。

【００３９】次に、エンジニアは、各ユニットモジュールに対してエイリアス決定テーブル（図３のエイリアス決定テーブル６０のような）を生成し、そのユニットモ

ジュールが帰属するユニットクラスに関連するフェーズクラスのそれぞれの一つに使用されるエイリアス名のそれぞれのエイリアス決定テーブルの定義を提供する。例えば、図2のオブジェクト階層に於いては、エンジニアは、充満、加熱、排出及び洗浄のフェーズクラスで使用されるエイリアス名のそれぞれの定義を含むように、ユニットモジュール反応器__01、反応器__02及び反応器__03のそれぞれに対してエイリアス決定テーブルを提供するであろう。図3に非常に良く示されているように、充満フェーズクラス56はフィード__タンクユニットクラス52にも関連しているので、フィード__タンクユニットクラス52に関連する何れかの他のフェーズクラスと同様に、(図3のフィード__タンク__06ユニットモジュール64のような)フィード__タンクユニットクラスに関連するユニットモジュールのそれぞれに対するエイリアス決定テーブルが、充満フェーズクラス56で使用されるエイリアス名のそれぞれに対する定義を含むことを確実にすることが、エンジニアに要求される。もちろん、上述のように、ユニットモジュールエイリアス決定テーブルの幾つかに於いては、エイリアス名の幾つかは「無視」によって定義され得る。なぜなら、それらは特定のユニットモジュールの動作に無関係だからである。

【0040】好ましくは、ワークステーション14内の構成アプリケーションは、各フェーズクラスに対する全てのエイリアス名がそのフェーズクラスが帰属するユニットクラスに帰属するユニットモジュールのそれぞれに対するエイリアス決定テーブルによってサポートされているかどうかを決定するために自動的にチェックを行うエイリアス定義チェックルーチンを含んでいる。一実施形態では、各ユニットクラスは、ユニットクラスに関連する全てのフェーズクラスで使用される全てのエイリアス名を含むエイリアス名のリストを提供するであろう。次に、チェックルーチンは、そのユニットクラスに関連する全てのエイリアス決定テーブルに於けるこれらのエイリアス名のそれぞれについて、正しいエイリアス定義が存在するかどうかが決定される。複数のユニットクラスが一つのフェーズクラスを共有しているので(図2及び図3の充満フェーズクラスに示されているように)、システム内で全体的に唯一に命名される必要がある異なるユニットクラスで、同じエイリアス名が使用され得る。他の実施形態では、チェックルーチンは特定のフェーズクラスに対するエイリアス名を決定し、そして、フェーズクラスのエイリアス名のそれぞれに対する正しい定義をエイリアス決定テーブルが含んでいるかどうかを決定するために、検査されたユニットクラスに関連する各ユニットモジュールに対するエイリアス決定テーブルをチェックする。このルーチンは、続けて次のフェーズクラスに進み、そしてこの操作を全てのフェーズクラスがチェックされるまで繰り返す。この操作の間、チェッキングルーチンは、そのユニットモジュールのためのエイリアス決定テーブルに基づいて各フェーズが決定されたかどうか、及びこのフェーズ即ちフェーズクラスが実行時間の動作に使用するためにコントローラ12にダウンロードされたかどうかを、フェーズテーブルに示して各ユニットモジュールのフェーズテーブルに書き込む。また、チェックルーチンは、各エイリアス名のそれぞれに対する定義が特定のエイリアス決定テーブルの何れかに存在するかどうか、そして、特定された定義が正しいか、即ちプロセス制御システム内の正しい位置又はデバイスを指しているかを決定し得る。このチェックは、システムハードウエア構成と実行時間の間にコントローラ12によって使用されたデータベースをミラーリングするようにセットアップされたワークステーション14内の構成データベースを使用して為される。このチェッキングルーチンの使用は、複数のユニットクラスによってサポートされるべきフェーズクラスを可能とするのに役に立つ。

【0041】フェーズテーブルは、実行時間の間に、フェーズクラスが適用され得るそれぞれの及び全てのユニットモジュールによってどのフェーズクラスがサポートされていないかを決定するためにエンジニアによって使用される。(そのユニットモジュールのエイリアス決定テーブルの不正な又は存在しないエイリアス定義により)一つのユニットモジュールによってさえフェーズクラスがサポートされていないとき、構成ルーチンは、決定されていないエイリアス定義によって停止又は中断するフェーズクラスに基づいて、コントローラ12が実行可能なルーチンを生成しようとするのを妨げるために、フェーズクラスがコントローラ12にダウンロードされるのを妨げるのが好ましい。更に、構成ルーチンは、フェーズテーブル内のダウンロードされたパラメータの設定に基づいたフェーズクラスのダウンロードを妨げ得る。

【0042】フェーズクラス及びエイリアス決定テーブルが全て適切に構成されたとき、それらはこれらのオブジェクトに基づいて実行時間の動作を行うことを可能とするためにコントローラ12にダウンロードされる。一般的に言えば、コントローラ12はエイリアス決定テーブルとその中にエイリアス名を有するフェーズクラスとをメモリ22内に格納する。しかしながら、フェーズクラス及びエイリアス決定テーブルは、所望なら、ワークステーション14の一つのメモリ20又は他のどのようなメモリにも格納され得る。いずれにしても、このようなルーチンが(ワークステーション14又はコントローラ12上に格納され実行され得る)バッチ実行ルーチンによって実際に必要とされ又は呼び出されるまで、コントローラ12は実行可能なユニットフェーズルーチンを生成することはない。バッチ実行ルーチンがバッチ実行を遂行するとき、それは、バッチプロセスが実行される

べき特定のユニットモジュールのそれぞれに対する各フェーズクラスのインスタンス化したものを最初に生成する。コントローラ１２（又はその中のプログラム）は、使用されるべきフェーズクラスにアクセスし、そのフェーズが実行されるべきフェーズクラスに関連するユニットモジュールについてのエイリアス決定テーブルをアクセスする。エイリアス決定テーブル及びフェーズクラスのためのＰＡＤを使用して、コントローラ１２は実行可能なユニットクラスを生成し、そこではＰＡＤ内のエイリアス名が決定され、又はエイリアス決定テーブル内のこれらの名称のための定義によって置き換えられる。もしフェーズクラスが一つ以上のＰＡＤを有しているなら、コントローラ１２は、ユニットフェーズを生成するのにどのＰＡＤが使用されるのかを決定するために、ユニットモジュールのＰＡＤ識別パラメータを使用する。その後、バッチ実行ルーチンによって指示されたように、ユニットフェーズ（即ち、フェーズクラスのインスタンス化されたバージョン）を実行する。

【００４３】コントローラ１２はそのメモリ２２内に（その中にエイリアス名を有する）フェーズクラスを格納しているので、いつでも各ユニットモジュール（即ち、全てのユニットフェーズの）に対する各フェーズクラスのインスタンス化された実行可能なバージョンをコントローラ１２が有していることは必要とはされず、このことは、コントローラ１２のメモリの要求を減少させる。事実、本発明に従えば、現在実行されているインスタンス化されたユニットフェーズのそれぞれとフェーズクラスのそれぞれを格納するのに十分なメモリのみを使用する。ユニットフェーズの実行後、コントローラ１２は格納されたフェーズクラスとそのユニットモジュールのための格納されたエイリアス決定テーブルからユニットモジュールの新たなユニットフェーズを生成することができるので、コントローラ１２はそのユニットフェーズを廃棄し得る。もちろん、もし、ユーザによってフェーズテーブルで定義されたとき、ユニットフェーズがコントローラ１２の動作に永続的にインスタンス化されるべきであるなら、そのユニットフェーズは廃棄されることなく、このユニットフェーズのために常に利用可能なメモリが存在することを確実にするためにコントローラのメモリ２２に保持される。いずれにしても、一般的なエイリアス化された制御ルーチン（フェーズクラス）をそれらが実際に必要とされるまで及び次にエイリアス決定テーブルを使用して実行可能な制御ルーチンを生成するまで格納することは、いつでもそれぞれのユニットモジュールに対する各フェーズクラスのための別々の実行可能なプログラムを格納するのにコントローラを必要とする従来技術のシステムに対して、必要とされるメモリの量を減少させる。しかしながら、実行時間の前に別々の実行可能な制御ルーチン（ユニットフェーズ）が生成されるので、コントローラ１２は、バッチ実行の前に解決している問題が一般的な制御ルーチンとエイリアス決定テーブルとの間に存在することを認識し、このことはバッチ実行を開始し、次に、決定できないエイリアス名によりその動作の間に停止されることを妨げ、このことは、実行時間の間の実行中にエイリアス名が決定される一般的な制御ルーチンが格納され実行される従来技術システムの抱える問題点である。

【００４４】実行可能なユニットフェーズは実行時間の前に生成されるので、そして、実行時間の間にコントローラ１２によって使用されるのはこのユニットフェーズであるので、一般的なフェーズクラスは常に利用可能であり、従って、このフェーズクラスは、それから生成された一つのフェーズユニットが実行されている間、他のユニットフェーズを生成するのに使用される。同様に、一般的なフェーズクラスは、そのフェーズクラスから発展したユニットフェーズが実行されている間にアップグレードされ又は変更され得、このことは、先のフェーズクラスから発展した現在実行中のルーチンを中断することなく、新たなフェーズクラスをユーザがダウンロードすることができることを意味している。このことは、現在実行しているプロセスを中断させることなく、コントローラ１２のアップグレードを可能とするので、有利である。

【００４５】更に、フェーズクラスは一以上のユニットクラスに関連づけられることができるので、単一のフェーズクラスが異なるタイプのユニット又はハードウエアに対して格納され使用され、このことは、、制御システム内の必要とされるオブジェクトの数を更に減少させ、コントローラ１２のメモリの要求を減少させる。また、フェーズクラスは、同じ（又は異なる）ユニットクラス内の異なる装置で使用され得る複数のＰＡＤを有し得るので、ユーザは、ハードウエアの些細な差異を賄うためにバッチ実行ルーチンをプログラムする必要はない。その代わり、フェーズクラスがこの差異を賄い、従って、たとえ異なるユニットモジュールがそれに関連するわずかに異なるハードウエアを有していても、バッチ実行ルーチンは、異なるユニットモジュール上の同じ機能を達成するために、同じフェーズクラスを呼び出し又は使用することができる。

【００４６】エンジニアによって発展させられてきたプロセス制御ルーチンは、或るパラメータ又は変数をも含み得、その値はインスタンス化されたプロセス制御ルーチン（即ち、ユニットフェーズ）がコントローラ１２内で生成された後に特定され得る。これらの動的に結合され又は動的に決定されるパラメータは、例えば、特定のユニットモジュール上で実行されるバッチのフェーズ内でユーザ又はバッチ実行ルーチンにとって異なる選択が可能であるときに、有用である。例えば、バッチ実行ルーチンには、使用されているレシピに基づいて図１の反応器＿０２のバルブ２０１又はバルブ２０２を開くかど

うかを決定することが要求される。例えば、もし、バッチ実行がビール等の炭酸飲料を作ることであるなら、そのバッチプロセスの充満フェーズの間反応器＿０２のバルブ２０１が開かれている必要がある場合である通常のビールを作るためにレシピが作成され、そのバッチプロセスの充満フェーズの間反応器＿０２のバルブ２０１が開かれている必要があるライトビールを作るためにレシピが形成される。これらの異なる充満動作のために（レシピに基づいて）２つの別々のフェーズクラスを有する代わりに、反応器＿０２ユニットモジュールに対するユニットフェーズがコントローラ１２内に生成された後、バッチが入力バルブパラメータを動的に特定するように実行可能とするのに有用である。

【００４７】上記のように、動的な結合を可能とする従来技術のシステムは、典型的にはアドレス配列を使用し、そこでは異なるポインタが制御ルーチン内で使用されている異なるアドレスに格納されることができ、そこでは各アドレスに関連する一つのポインタが存在する。しかしながら、これらのアドレス及びそこでのポインタの追跡を続けることは困難であり、動的結合を形成しようとする前に、そのアドレスでのポインタが正しいかどうかを動的に決定する方法は存在しない。もし、ポインタが正しくなければ、制御ルーチンは典型的には停止し、この停止は一般的には生産時間と材料のロスを招くので特にバッチ実行の途中に於いては非常に好ましくなく、追跡中にバッチ実行を戻すには非常に困難なオペレータ介入を必要とする。

【００４８】本発明によれば、何れのフェーズクラスについてのプロセス制御ルーチンに於いても使用されるように、（動的に特定された）変数又はパラメータ（動的参照パラメータ）を動的に結合することを可能とするのが好ましい。換言すれば、幾つかのケースの場合には、フェーズクラスのＰＡＤに於ける動的参照パラメータを、実行可能なユニットフェーズがフェーズクラスから生成されたときにユニットフェーズに変換され又は持ち越された動的参照パラメータで置き換えることが好ましく、そこでは、この動的参照パラメータの値は、ユニットフェーズが生成された後、そしてユニットフェーズが実行を開始されたとき（即ち、実行時間の間）に於いてさえ、特定されることができる。上述のように、このパラメータは、例えば、パラメータの値についての選択の決定が、構成時間に於いて利用可能ではない情報に基づいているとき、例えばオペレータの入力に基づく選択、バッチ実行ルーチンから通過してきたレシピパラメータに基づく選択、制御変数の実行時間値に基づく選択等のときに、有用である。

【００４９】プロセス制御ルーチンに於いて使用するための動的参照パラメータは、動的参照パラメータはコントローラ１２のメモリ２２（又は所望なら幾つかの他のメモリ）に格納されることを除いて、プロセス制御システム内で他のハードウエア又はソフトウエアのパラメータを定義するのに使用される約束事を使用して定義される。

【００５０】動的参照パラメータは、その動的参照パラメータを使用して実行される異なる動作を可能とするために、複数の属性又はフィールドを有していることが好ましい。特に、動的参照パラメータは、以下の表１に定義するフィールド又は属性を含み得る。

【００５１】

【表１】

名称	タイプ	構成の可能性	読み出し可能性	書き込み可能性
DREF	文字列	不可	文字列としての読み出しは、可能なら決定されたエイリアス名を伴う現在の参照パスに戻る。浮動小数点、整数などとして読み出すことは不可能	文字列としての書き込みは新たな参照パスを確立する：理想的には結合されるべき新たな外部参照を引き起こす。浮動小数点、整数などとして書込みは不可能
CONN	整数	不可	DREFフィールドの参照の結合状態を定義する。整数としての読み出し:0は参照が確立されたことを意味する。-1は参照が決して開始されないことを意味する。>0は結合が進展し後に再チェックする価値がある	不可
DRFV	浮動	不可	参照フィールドの値は浮動小数点の値として解釈される。数字の表現として使用される。もし参照フィールドが浮動小数点として解釈されないなら（例えば文字列型フィールド）、読み出される値はMAX＿値である。また、もしCONN属性の値が0に等しければMAX＿値として読まれる。	この属性への書き込みは参照フィールドの値を浮動小数点値として非同期的に書かれたものとする。数字の表現として使用される。もし、参照フィールドが浮動小数点値として解釈され得ないとき(例えば文字列型フィールド)、書き込み動作は失敗する。もしCONN属性の値が0に等しくなければ書込まない
DRSV	文字列	不可	参照フィールドの値は文字列と解釈される。文字列の表現として使用される。もし参照フィールドが数字ならその数値は（可能なら）文字列型に変換される。もしもしCONN属性の値が0に等しくなければ、空の文字列が返される。	これへの書き込みは、参照されたフィールドの値を非同期的に文字列として書かれたものとする。文字列型の表現で使用する。もし、参照フィールドが数値なら、文字列は（可能なら）数値の形式に変換され書き込まれる。もしCONN属性の値が0に等しくなければ、書込まない。
DRST	整数	不可	（もしあれば）参照パラメータの状態フィールドの値を読み出す。連続する状態に富能する計算で使用される、動的参照フィールドからの状態値のローカルパラメータ状態フィールドへのコピーにはこの属性を使用する。もし、CONN属性の値が0に等しくなければ、MAX＿値に設定する。もし動的参照が状態フィールドを有していないなら「良好」の状態をセットする。	不可
WRST	整数		最後の動的参照フィールドへの書き込み動作の成功を示す参照フィールドの状態値を読み出す。値は、例えば以下の何れかによって区別される：「進展中」「成功」「conn失敗」「不良」等。新たな参照文字列がDREF属性に書き込まれたとき、この属性は「不良」にリセットされる	不可

【0052】DREF（動的参照）属性の値はポインタであり、その動的参照が現在関連しているパラメータ又はフィールドへのタグ又はパスである。DeltaVシステムでは、このポインタは、デバイス又はモジュールパラメータのようなモジュールを指す（パスなどの）文字列値である。例えば、「出力＿POS」の動的参照パラメータは、例えば、命令
'OUTLET#POS.DREF':= "VLV1004/AO/OUT.CV"　又は'OUTLET#POS.DREF':= "";//空の文字列.
を使用するパラメータ／フィールドへの参照である文字列定数に割り当てられることができる。もちろん、他のシステムでは、DREFのフィールドの値は、そのシステムが装置、モジュール等を特定する方法に依存する数字又は文字列であり得る。一般的に言えば、DREF割当の使用は、動的参照は動的参照パラメータのDREFフィールドへのフルパラメータ参照文字列パスを割り当てることにより確立されるので、コントローラ12に新たな動的参照パラメータを生成する。制御動作の動的目標が必要とされるとき、ユーザは、実行時間までに決定され得ない各パラメータの動的参照パラメータを生成することが必要となる。例えば、ユニットタグクラス、ユニットクラスの一部分、フェーズクラスにおけるフェーズアルゴリズムパラメータ、及びフェーズ論理モジュールに於けるモジュールレベルパラメータとしての場合を含めて、そのシステムに生成される他の何れの外部参照パラメータの状況に於いても、動的参照パラメータが生成され得る。もちろん、動的参照パラメータは、他の状況に於いても使用され得る。他のユーザが生成したパラメータについても、ユーザは、パラメータ命名構成ルールが許容する方法で、ローカル名称の範囲（即ち、同じレベルで定義された他のパラメータ、ブロック、又はフェーズクラス名称と同じ名称を有することはできない）で唯一の方法で、動的参照パラメータの名称を構成する。DREF属性への文字列の書き込みは高価な操作であり（先の外部参照オブジェクトを破壊し、新たな外部パラメータを構築する）、従って一般的には繰り返しでない（パルスの）表現に於いて行われるべきである。DREF属性への新たな文字列の書き込みは、即座にCONN属性（及び動的参照パラメータの他の属性）からの読み出し値の「不良」への変更を生じさせ（もし結合が即座に確立されなければ）、引き続くこれらのフィールドのテストの表現が正確に機能する。

【００５３】ＤＲＥＦ属性への割当は、例えば連鎖オペレーション（２又はそれ以上の文字列の連鎖）、文字列選択オペレーション（多くの可能性のある文字列の一つがオペランドの値に基づいて選択される）、又は他の何れの文字列オペレーションをも含めて、所望の文字列オペレーションを使用して達成される。

【００５４】ＣＯＮＮ（結合）属性は、ＤＲＥＦ属性によって特定された値が正しいか又は制御システムの状況下の決定可能なフィールドかについての指標を提供する。ＤＲＥＦ属性が変更され又は設定されたとき、コントローラ１２は、それようのタグ又はパスが存在するか及び制御システム１０の現在の構成に位置し又は決定されているかを見るために、即座に及び自動的にその値をテストする。もし、ＤＲＥＦ属性の値が正しく決定可能であるなら、ＣＯＮＮ属性は０に設定される。しかしながら、もしＤＲＥＦ属性の値が決定不可能であり、例えば制御システムの状況下で不適切であり若しくはちょうど存在していないことにより決して決定され得ないなら、ＣＯＮＮ属性は－１に設定される。もし、コントローラ１２がＤＲＥＦ属性を決定しようと積極的に試みても、接続動作などによる待ち時間のためにそのようにすることができないなら、ＣＯＮＮ属性の値は０より大きい値に設定され、この値は未だ決定されておらず後に決定されることを示している。もしＤＲＥＦ属性が未だ決定されていないなら、時間切れの期間の後、ＣＯＮＮ属性の値は－１に設定されることが好ましい。ＣＯＮＮ属性は、実行時間の間に動的参照のテストを可能とするので、非常に有用である。例えば、ＤＲＥＦ値が正しく定義されている場合にのみ、単に「IF < dynamic parameter name > .CONN= 0, THEN < action to be taken > 」という命令はアクションを起こすのに使用されることができる。この可能性は、制御ルーチンが動的参照を使用し得るように、しかしもし動的参照が実行時間に不正であるなら中断しないように書かれることを可能とする。もちろん、分岐命令、停止又は中断命令等の、ＣＯＮＮ属性を使用する他のどのようなテスト又は命令も使用され得る。更に、ＣＯＮＮ属性は、接続の成功又は失敗を示すために、（０，１及び０より大きい）所望の他の値を採り得る。

【００５５】動的参照パラメータのＤＲＦＶ（動的参照浮動値）読み取り／書き込み属性は、浮動小数点又は整数の値として、ＤＲＥＦ属性によって特定されるフィールドへの読み出し及び書き込みを可能とするのに使用される。一実施形態では、ＤＲＦＶ属性は、もしＣＯＮＮ属性が０でなければ、最大値又は他の特定された値に設定される。また、この実施形態では、ＤＲＥＦ属性は、もしＣＯＮＮ属性が０でなければ、動的に参照されるフィールドへの書き込みを妨げるであろう。同様に、動的参照パラメータのＤＲＳＶ（動的参照文字列値）読み取り／書き込み属性は、ＤＲＥＦ属性によって文字列値として特定されるフィールドへの読み出しと書き込みを可能とするのに使用される。一実施形態では、ＤＲＳＶ属性は、もしＣＯＮＮ属性が０でなければ、空の文字列に設定され、もしＣＯＮＮ属性が０でなければ、書き込みを妨げる。これらは、動的参照フィールド実際に存在するか又は正しいかを示す文字列又は数値のフィールドの両方として動的参照フィールドから書き込まれ及び読み出されることが可能であるので、これらは有用な属性である。もちろん、ＤＲＥＦ若しくはＤＲＳＶ属性又は他の特別に生成された属性も、ＤＲＥＦ属性によって特定されたフィールドへのブールの値又は（配列で格納された値のグループのような）配列値を読み出し及び／又は書き込むのに使用されることができる。

【００５６】ＤＲＳＴ（動的参照状態）属性は、ＤＲＥＦ属性によって特定されたフィールドに関連する状態属性の読み出しを可能とする。DeltaV及びフィールドバスプロトコル等の或るコントローラ又は通信プロトコルでは、幾つかのパラメータ又はフィールドは、その値が良好か不良か不確定かどうかなどを等の値の状態を示す値又は状態を含んでいる。ＤＲＳＴ属性は、動的に参照されるパラメータのこの状態値にアクセスすることを可能とする。ＷＲＳＴ（書き込み状態）属性は、ＤＲＥＦ属性によって指定されたフィールドの書き込み状態値を読み取る。この状態は、ＲＥＦ属性によって指定されたフィールドへの最後の書き込みオペレーションの成功を示し、動的参照フィールドの書き込み状態へのアクセスを提供する。

【００５７】もちろん、所望なら、他の属性は、そのモード、動的参照フィールドに関連する制限又は他の状態への読み出し又は書き込みを提供すること、又は動的参照フィールドのどのような属性への他の読み出し又は書き込みを実行する等の他のオペレーションへの動的参照パラメータが備えられ得る。同様に、ここに確認した属性は、接続の、又は読み出し若しくは書き込みオペレーションの成功、失敗等を示す他の名称又は値を採り得る。

【００５８】動的参照パラメータがユニットモジュールの状況で使用されるとき、即ち、その中に動的参照参照を有するユニットフェーズの生成に際して、ＤＲＥＦ属性に書き込まれる文字列はエイリアス名について検査され、そして何れかのエイリアス名がそのユニットモジュールの現在のエイリアス決定テーブルに基づいて置換される。結果として、動的参照はエイリアス名を使用してフィールドを特定するように生成され得、そして、ユニットフェーズがフェーズクラスから生成されたとき、これらのエイリアス名がなお決定される。このことは、たとえ動的参照が実行時間までに決定されなくても、又は動的参照パラメータのＤＲＥＦ属性への書き込みに基づいて実行時間の間に決定されても、プロセス制御ルーチンに動的参照パラメータがより広く使用されることを可

能とする。

【００５９】ＳＦＣアルゴリズムを使用するとき、動的参照パラメータを介する書き込みは、ＳＦＣの設計に依存して幾つかの方法で為され得る。例えば、（所望なら、書き込みオペレーションの確認がＳＦＣの後の部分で論理によって扱われると仮定して）そのルーチンは所望の値をステップ表現のステートメントとして丁度割り当て、そのルーチンは一度書き込み及びＷＲＳＴ属性が「実行中」の値以外の値になるまで休止するのにパルス／割当タイプのアクションを自信を持って使用し、又はそのルーチンはその値が達成され又はステップ時間が長すぎることを遷移表現が検出するまで繰り返し値の書き込みを行うように非格納／割当タイプのアクションを使用する。このように、ＳＦＣアルゴリズムを使用するとき、パルス／割当タイプのアクションの使用を介して動的参照を確認しながら確立し検証し、これにより、アクセス表現がフル動的参照パスを決定し及び適当な動的参照パラメータのＤＲＥＦフィールドに（モジュールレベル又はフェーズレベルで）それを割り当て、そして０（接続され又は決して接続されない）又はそれより小さいＣＯＮＮ属性の値について確認表現をテストする。それに代えて、読み出しを意図する動的参照パラメータについては、ＤＲＦＶ属性の値は、妥当性のある値（最大_値に対するものとして）について読まれてチェックされ、確認時間切れ値が最小の秒数（例えば５秒）に設定され得る。

【００６０】更に、もし幾つかの動的参照パラメータがアルゴリズムの同じポイントに確立されたら、同じＳＦＣステップに於けるそれぞれについてアクションを生成するのが好ましい。次に、ＳＦＣ遷移の次の表現に於ける"'RESOLVE STEP/PENDING CONFIRMS.CV' = 0"のような単一の項目は、アルゴリズムが全ての動的参照パラメータがそれらの最終状態に於いて「確立された」を有するまで先へ進むのを妨げることができる。もし、アルゴリズムがエイリアス接続が存在しない又は無視を取り扱わなければならないなら、アルゴリズムの実行をガイドするために、これに続く表現は個々の動的参照パラメータのＣＯＮＮ属性をテストすることができる。

【００６１】一度動的参照が確立され（即ちＤＲＥＦ属性が書かれると）動的参照が検証されると（ＣＯＮＮ属性が０であると）、ＤＲＦＶ属性、ＤＳＲＶ及びＤＲＳＴフィールドには、参照パラメータに於ける値が設定されるように書き込まれる。（FAIL#MONITOR複合ブロック等の連続して実行される）連続するアルゴリズムに於いては、（既に確立されている）動的参照パラメータを介する読込への推奨されるアプローチは、

```
IF ('OUTLET#POS.DRFV' != <desired value>) AND
('OUTLET#POS.WRST' != <in progress value>) THEN
'OUTLET#POS.DRFV' = <desired value>;
```

のような形式が採られ、これは、最後の試みが未だ実行中である時に書き込みを避けるような方法で達成されるまで、所望の値に参照パラメータを押し進めようと連続的に試みる。

【００６２】本発明は特定の実施形態に関連して記述されているが、これは例示のみを意図し、本発明を限定するものではなく、本発明の精神及び範囲を逸脱することなく、記載されている実施形態に改変、付加又は削除が行なわれ得ることは当業者に明らかであろう。

【００６３】

【発明の効果】実行時間の前に決定されたエイリアス名を使用するプロセス制御ルーチンと、実行時間の間に決定される動的参照パラメータのような間接参照は、所望のプロセス制御プログラミング環境内で使用され実行され、どのような所望のタイプのプロセス制御通信プロトコルを使用するどのようなプロセス制御システムに於いても使用され、更に、どのようなデバイス又はデバイスのサブユニットに関連するどのようなタイプの機能を実行するのにも使用され得る。ここに記載したような間接参照を使用するプロセス制御ルーチンは、例えばコントローラ又は他のプロセス制御デバイスに格納されたソフトウエアで実行されるのが好ましい。しかしながら、これらのルーチンは、これに代えて又はこれに加えて、所望のハードウエア、ファームウエアなどで実行され得る。もしソフトウエアで実行されるなら、個々で議論したプロセス制御ルーチンは、磁気ディスク、レーザディスク若しくは他の格納メディア、又はコンピュータのＲＡＭ若しくはＲＯＭ等に格納され得る。同様に、このソフトウエアは、例えば電話回線、インターネット等の通信チャンネル上を含む公知の又は所望の配送手段を介してユーザ又はデバイスに配送され得る。

【図面の簡単な説明】

【図１】プロセス装置の制御を実行するためのエイリアス名及び／又は動的参照パラメータを有する押出し成型装置の制御ルーチンを使用するプロセス制御ネットワークの部分的ブロックダイヤグラムである。

【図２】図１のプロセス制御ネットワークの概念的構成を示すオブジェクト構造を示すブロックダイヤグラムである。

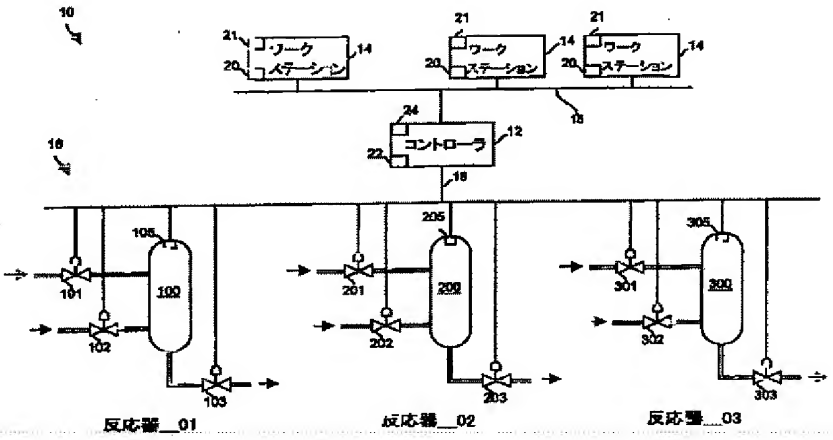
【図３】図２のオブジェクト構造の一部を拡大したブロックダイヤグラムである。

【符号の説明】

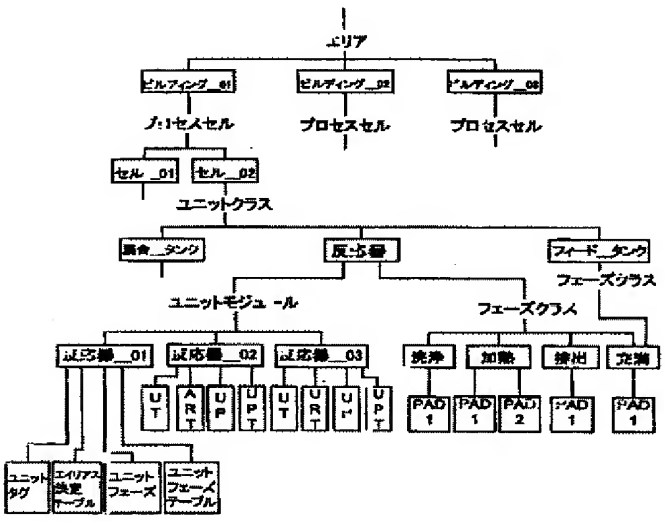
１０　プロセス制御ネットワーク
１２　プロセスコントローラ
１４　ワークステーション
１５　イーサネット接続
１５　イーサネット通信ライン
１６　プロセス
１８　バス
２０，２２　メモリ
２１，２４　プロセッサ

５０　反応器ユニットクラス
５２　タンクユニットクラス
５２　ユニットクラス
５４，５４，６４　ユニットモジュール
５６，５８　フェーズクラス
５６　充満フェーズクラス
５８　排出フェーズクラス
６０　エイリアス決定テーブル
６２　フェーズテーブル
１００，２００，３００　反応容器
１０１，１０２　入力バルブ
２０１，２０２　入力バルブ
３０１，３０２　入力バルブ
１０３，２０３，３０３　出力バルブ
１０５，２０５，３０５　デバイス
１０５，２０５　流体レベルメーター

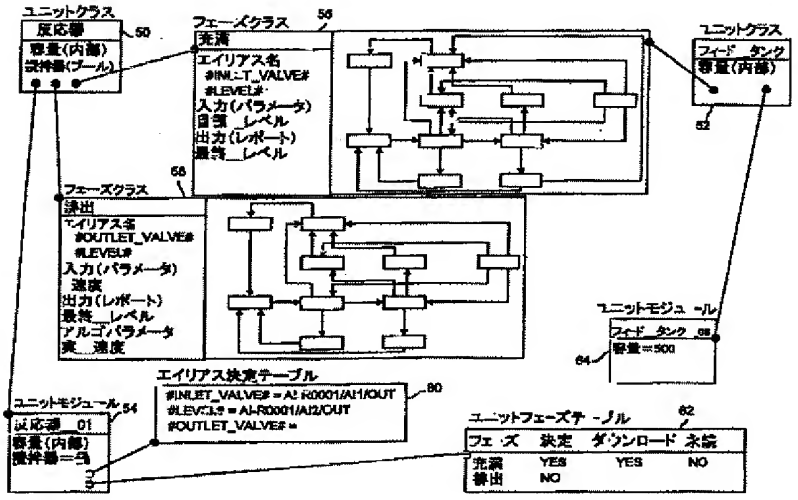
【図１】



【図２】



【図3】



フロントページの続き

(72)発明者 ヘイブコスト, ロバート ビー.
アメリカ合衆国 78728 テキサス オースティン クリスタルコート 14507

(72)発明者 スティーブンソン, デニス エル.
アメリカ合衆国 78681 テキサス ラウンド ロック セイバートゥース ドライブ 16904

(72)発明者 デイツ, デビット エル.
アメリカ合衆国 78681 テキサス オースティン マウンテン ビラ ドライブ 5915